

版权注意事项：

- 1、书籍版权归作者和出版社所有
- 2、本PDF仅限用于个人获取知识，进行私底下的知识交流
- 3、PDF获得者不得在互联网上以任何目的进行传播
- 4、如觉得书籍内容很赞，请购买正版实体书，支持作者
- 5、请于下载PDF后24小时内删除本PDF。



- ◆ 来自资深软件开发工程师及培训讲师的Java报表课程
- ◆ 全面介绍JasperReports+iReport报表工具及技术的使用方法
- ◆ 解决报表开发问题，为多样化报表提供解决方案

JasperReports + iReport

报表开发详解

高洪岩 编著



清华大学出版社



JasperReports + iReport

高洪岩 编著

报表开发详解

清华大学出版社
北京

目 录

第 1 章 初识 JasperReports	1
1.1 JasperReports 的基础知识	1
1.2 第 1 个 JasperReports 打印示例——中文静态文本	2
1.2.1 创建模板文件	2
1.2.2 在 Web 项目中以 PDF 文件显示出来	8
1.2.3 利用程序将 .jrxml 导出为 .jasper 文件并用 PDF 显示	13
1.3 填充报表数据——使用 Map 参数	15
1.3.1 新建报表模板文件	15
1.3.2 创建传递参数的 Servlet 对象	17
1.3.3 显示效果	19
1.3.4 打印 List 中 Userinfo.java 实体类示例	20
1.4 填充报表数据——使用 JDBC 向导作为数据源	23
1.4.1 新建报表 JDBC 数据源	23
1.4.2 新建报表模板文件	24
1.4.3 设计报表	26
1.5 使用向导分组显示数据	29
1.6 在 iReport 中使用表达式 Expression	32
1.7 将报表导出为 PDF 文件	34
1.8 报表的常用属性	38
1.8.1 分栏分列的效果	39
1.8.2 Title 和 Summary 在单独的页面打印	41
1.8.3 多列横向与纵向排序打印效果	42
1.8.4 Summary with Page Header and Footer 属性	43
1.8.5 Float column footer 属性	45
1.8.6 When No Data 属性	47
1.9 各个 Band 存在数据时的打印效果	50

第 2 章 控 件.....	54
2.1 控件的常用知识	54
2.2 控件的对齐	57
2.3 控件的常用属性	64
2.3.1 Forecolor、Backcolor、Opaque 属性	64
2.3.2 Blank When Null 属性	65
2.3.3 Position Type 属性	67
2.3.4 Stretch Type 属性	74
2.3.5 Print Repeated Values 属性	81
2.3.6 Remove line when blank 属性	83
2.3.7 Print In First Whole Band 属性	84
2.3.8 Print When Detail Overflows 属性	86
2.4 控件的使用方法	88
2.4.1 形状控件	88
2.4.2 Image 控件	89
2.4.3 Image 控件	97
2.4.4 文本控件	106
第 3 章 Fields、Parameters、Variables 对象及 Group 分组	109
3.1 Fields 对象的使用	109
3.1.1 使用 Text Field 控件显示数据表字段值	110
3.1.2 使用 Fields 结合 JDBC 的 Connection 对象显示值	112
3.1.3 使用 Fields 对象显示 Java 集合中实体类的属性值	113
3.2 Parameters 对象的使用	115
3.2.1 使用 Parameters 动态生成 userid 值	116
3.2.2 使用 Parameters 动态生成 Date 区间的测试	118
3.2.3 使用 Parameters 动态生成 where 语句	121
3.2.4 使用 Parameters 对象实现 SQL 的 IN 及 NOTIN 查询	123
3.3 Variables 对象的使用	125
3.3.1 Calculation 属性	127
3.3.2 Evaluation Time 属性	132
3.3.3 Increment type 属性	137
3.4 Group 分组的使用	144
3.4.1 Group 分组的使用方法	144

3.4.2	Group 分组的常用属性	149
3.4.3	Group 分组中的 Print When Group Changes 属性	154
3.4.4	Group 分组中的 Reset type 属性	157
3.5	常用小实验	163
3.5.1	实验 1	163
3.5.2	实验 2	165
3.5.3	实验 3	167
3.5.4	实验 4	172
3.5.5	实验 5	173
3.5.6	实验 6	174
3.5.7	实验 7	175
3.5.8	实验 8	176
第 4 章	字体 Font、样式 Style 及模板 Templates	179
4.1	字体 Font	179
4.1.1	使用自带字体	180
4.1.2	使用第三方字体	181
4.2	样式 Style	186
4.2.1	创建样式 Style	186
4.2.2	创建条件样式 Conditional Style	188
4.2.3	创建通用样式 Style	191
4.3	模板 Templates	194
第 5 章	子报表 Subreport	197
5.1	子报表 Subreport 的基础知识	197
5.1.1	子报表 Subreport 的 .jasper 文件来源	198
5.1.2	子报表 Subreport 的示例——静态文本	199
5.1.3	子报表 Subreport 的示例——动态数据 JDBC	207
5.1.4	子报表 Subreport 的示例——打印实体类中的 List<Userinfo>	216
5.2	子报表 Subreport 的参数传递	221
5.2.1	从 Servlet 传递一个 Map 类型的参数到子报表	221
5.2.2	对表达式进行计算后再传入子报表	225
5.2.3	对子报表传递 List<Userinfo>数据源	226
5.2.4	示例: 从主报表中取得子报表返回的参数值	230

第 6 章 图表 Chart	238
6.1 图表 Chart 的使用——饼状图	238
6.1.1 新建 JavaBean 数据源的报表模板	238
6.1.2 配置 Chart	239
6.1.3 创建 Servlet 对象	241
6.1.4 运行效果	242
6.1.5 图表 Chart 的常用属性——饼状图	242
6.1.6 图表 Chart 的常用选项——饼状图	248
6.2 图表 Chart 的使用——柱状图	266
6.2.1 使用柱状图显示报表	266
6.2.2 图表 Chart 的常用属性——柱状图	269
6.3 Chart 图表的使用——曲线图	280
6.3.1 使用 JDBC 数据源	280
6.3.2 使用 JavaBean 数据源	281
6.4 在图表 Chart 中添加超链接	284
6.5 在图表 Chart 中使用皮肤 Themes	288
第 7 章 数据集 Dataset、List 控件 及 Table 控件	292
7.1 数据集 Dataset	292
7.1.1 创建核心 Servlet	293
7.1.2 创建报表模板	294
7.1.3 创建 Dataset 数据集	294
7.1.4 配置 Dataset 数据集	295
7.1.5 关联 Dataset 数据集	297
7.2 List 控件	299
7.3 Table 控件	303
7.3.1 使用 Table 控件	303
7.3.2 合并单元格	309
7.3.3 使用 JavaBean 作为报表的数据源	311
第 8 章 实用技巧	317
8.1 导出各种文件格式	317
8.1.1 导出.xls 文件	323
8.1.2 导出 PDF 文件	328
8.1.3 导出 DOC 文件	330

8.1.4 导出 HTML 文件	331
8.2 取消报表分页	339
8.3 实现当前页/总页数的效果	339
8.4 巧用 Text Field 控件的 Borders 属性	340
8.5 一次输出多个报表	342
8.6 静态文本多行显示	345
8.7 设计带边框的表格	346

第2章 引言

作为本书的第1章,为了让读者快速入门学习报表的相关内容,特设置本章介绍报表报表表示例来作铺垫,首先掌握以下几个知识点:

- ★ 了解 JSP 型报表生成过程的步骤与相关过程
- ★ 使用 JSP/JspWriter 类输出打印报表
- ★ 使用 Parameters 参数对象
- ★ 导出 PDF 文件
- ★ 使用报表常用属性

1.1 JasperReports 的基础知识

在 Java 开发领域,处理报表打印模块的框架层出不穷,其中使用最广泛的是 JasperReports。具体来讲它并不是一个独立的应用程序,它通常被设计为 JSP 类库包使用,如 JavaSE 或 JavaEE 项目,或常见的 Web 项目都可以使用它。

JasperReports 是用 Java 编写的,所以有使用到 Java 的地方就可以使用 JasperReports,而且它还有跨不同平台的版本,如 Linux、Mac 操作系统上都可以使用。使用 JasperReports 只需要一个类似于 JasperReportxx.x.jar 格式的 jar 文件就可以了。本教程使用 jasperreports-4.6.0.jar 版本来进行学习。虽然使用 JasperReports 是如此简单,添加一个 jar 文件就可以了,但如果想实现一些高级的功能,如连接数据库、XML 解析、日志处理、生成 PDF 文件等,那就需要依赖于其他的 jar 包文件了,并且要添加到项目中或 classPath 路径中。较为欣喜的是,从官方网站下载的 jasperreports-4.6.0-project.zip 文件中有所有程序员需要用到的资源,如文档、使用手册等。解压效果如图 1-1 所示。

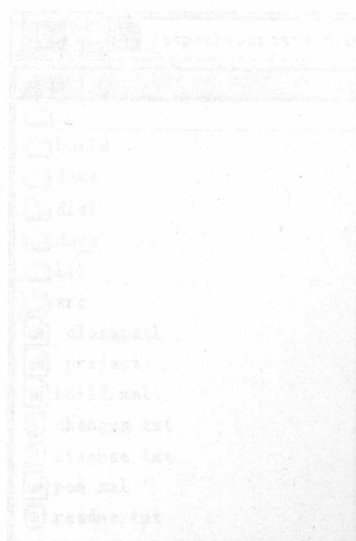


图 1-1 压缩包中的内容

第 1 章 初识 JasperReports



导言

作为本书的第 1 章，为了让读者快速进入学习报表的状态，将会以若干个应用型的报表示例来作演示，重点掌握以下几个知识点：

- ★ 用 iReport 创建报表模板的步骤与配置过程
- ★ 使用 SQL/JavaBean 数据源打印报表
- ★ 使用 Parameters 参数对象
- ★ 导出 PDF 文件
- ★ 使用报表常用属性

1.1 JasperReports 的基础知识

在 Java 开发领域，处理报表打印模块的框架数不胜数，但其中比较著名的是 JasperReports，具体来讲它并不是一个独立的应用程序，必须要嵌入到 Java 应用程序中运行，如 JavaSE 或 JavaEE 项目，或常见的 Web 项目都可以使用它。

JasperReports 是用 Java 编写的，所以有使用到 Java 的地方就可以使用 JasperReports，而且它还有跨不同平台的版本，如 Linux、Mac 操作系统上都可以使用。使用 JasperReports 只需要一个类似于 JasperReportsx.x.x.jar 格式的.jar 文件就可以了。本教程使用 jasperreports-4.6.0.jar 版本来进行学习。虽然使用 JasperReports 是如此简单，添加一个.jar 文件就可以了，但如果想实现一些扩展的功能，如连接数据库、XML 解析、日志处理、生成 PDF 文件等，那就要依赖于其他的.jar 包文件了，并且要添加到项目中或 classPath 路径中，较为欣喜的是，从官方网站下载的 jasperreports-4.6.0-project.zip 文件中有所有程序员需要用到的资源，如文档、使用手册等，解压效果如图 1.1 所示。

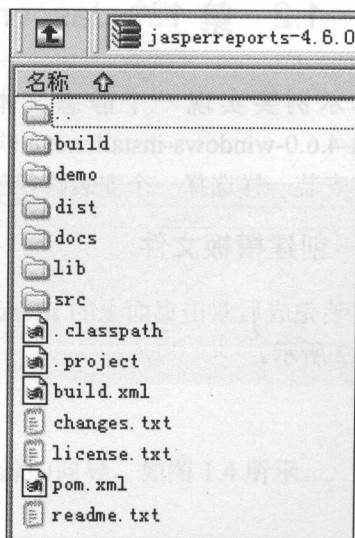


图 1.1 压缩包中的内容

在学习 JasperReports 框架技术时需要知道如下几点：

- 若想让 PDF 支持中文，需要有中文语言的 JAR 包文件。
- 若想生成不同格式的报表文件，需要调用不同的导出类。
- 在生成 HTML 格式报表文件时，需要注意图片路径。
- 在报表使用所有数据源的种类中，最灵活的就是 JavaBean 方式，该方式易于控制，扩展性好。
- 不同数据源之间的 jrxml 模板文件不通用。
- 设计基于 JavaBean 的数据源方式时，需要在 iReport 软件中的 Classpath 选项中配置 class 文件所存放的路径。



提示

需要说明的是，限于篇幅，本书不可能把 iReport 这个软件的所有功能和操作都记录下来，因为它是一个工具软件，就像 Photoshop 一样，总有人认为某些功能重要，某些功能不重要，所以本书只对开发中常见的问题进行详细介绍，如果想更多地了解报表的细节，请参考相关的帮助文档。

JasperReports 的模板是定义一个指定打印格式的相对通用的文件，在这个文件中可以定义一些元素，如页头、内容、页脚、分组、分栏等信息，通过将数据传给模板文件，就可以打印大段的文本（包括数据表中的数据），甚至还可以打印图片。

JasperReports 的模板是一个 XML 文件，扩展名为 .jrxml，在使用时需要将 .jrxml 文件编译成 .jasper 文件，可以手动设计这个 .jrxml 文件，当然这种方式效率比较低，所以与 JasperReports 配套的还有一个组件——iReport，它在 Windows 平台下是以 .exe 文件作为安装文件，它的主要作用就是以图形化的设计方式来创建 JasperReports 模板文件。

1.2 第 1 个 JasperReports 打印示例——中文静态文本

本示例要实现一个静态文本——Hello World 的打印，所以前期的工作是先把 iReport-4.6.0-windows-installer.exe 软件安装到系统中，此文件的安装过程非常简单，就像普通的软件安装一样选择一个安装路径就可以了，不需要一些额外的配置。

1.2.1 创建模板文件

安装完成后双击桌面上的 iReport-4.6.0 快捷方式，进入 iReport 软件，它的软件欢迎界面如图 1.2 所示。



图 1.2 进入 iReport 前的欢迎界面。

成功进入软件后，软件界面如图 1.3 所示。

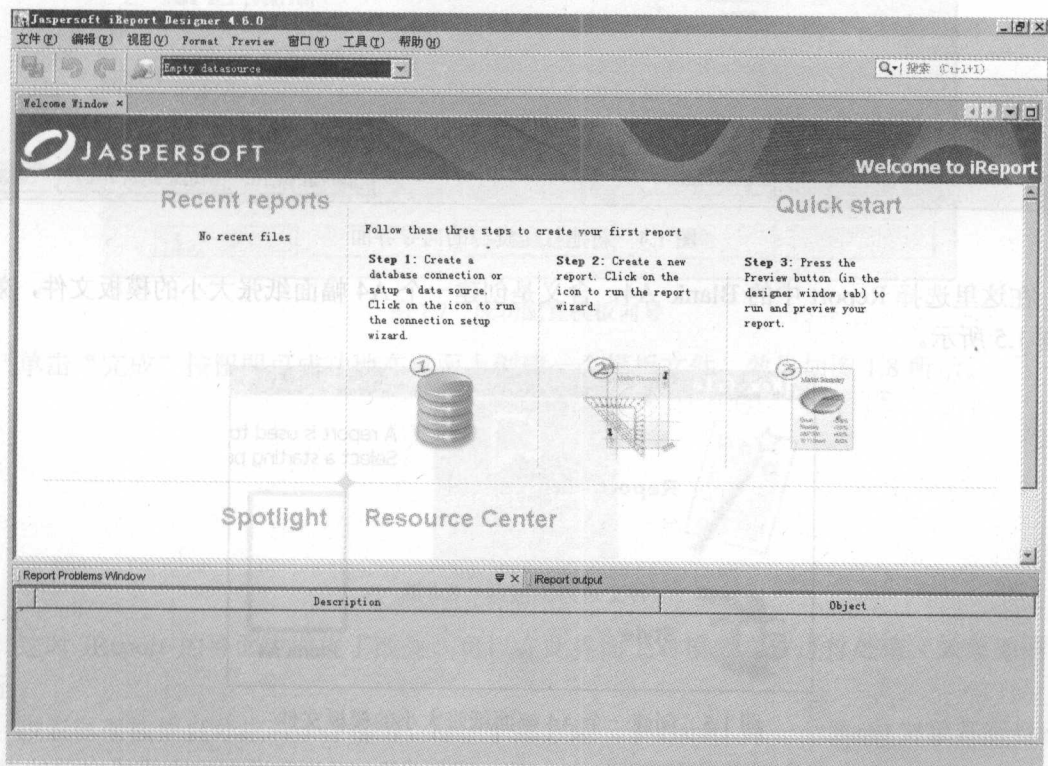


图 1.3 iReport 软件的界面

选择“文件”→New 命令，弹出一个新建打印有关资源的向导，如图 1.4 所示。

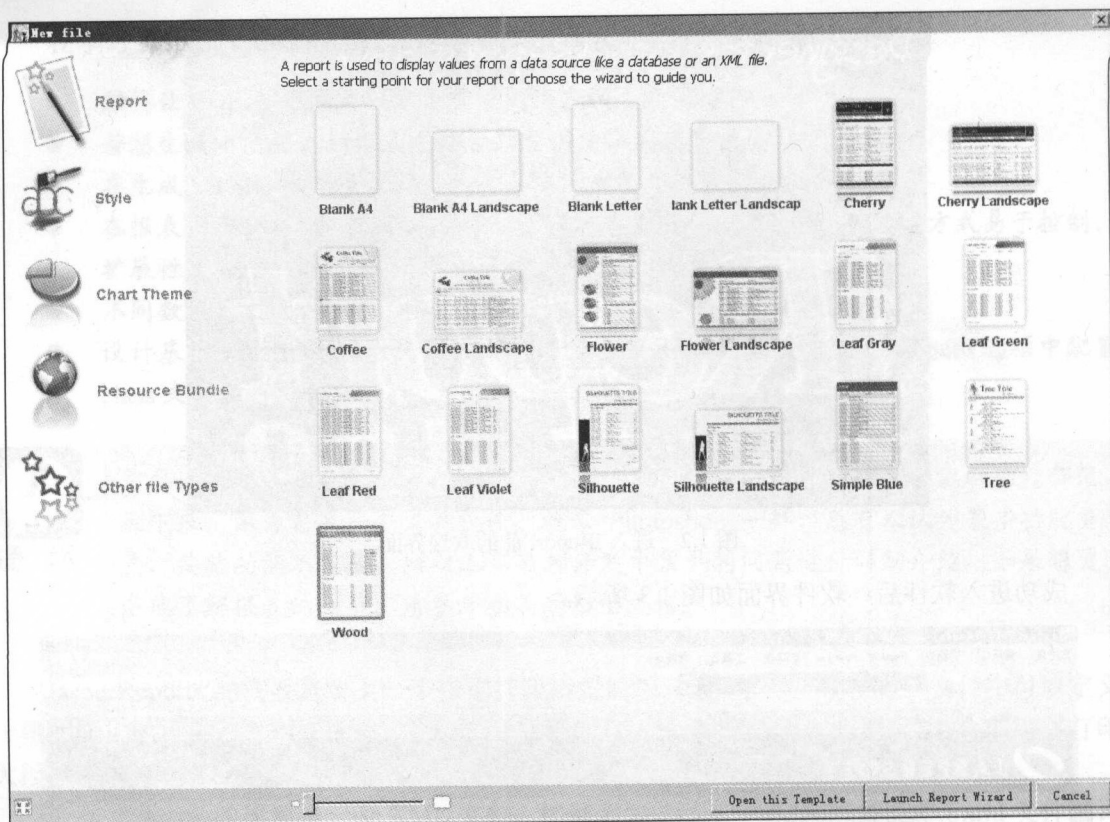


图 1.4 新建打印资源的向导界面

在这里选择 Report 中的 Blank A4，含义是创建一个 A4 幅面纸张大小的模板文件，效果如图 1.5 所示。

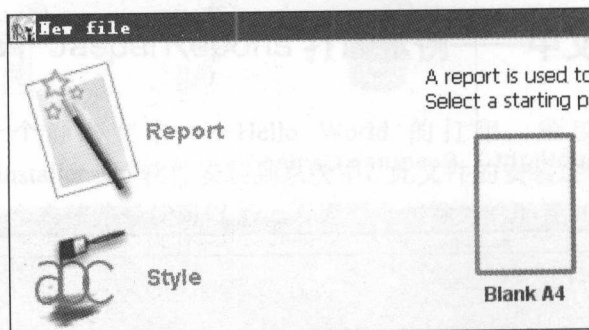


图 1.5 创建一个 A4 幅面纸张大小的模板文件

选中后单击右下角的 **Open this Template** 按钮，即打开这个模板，并且设置保存模板文件的路径，效果如图 1.6 所示。

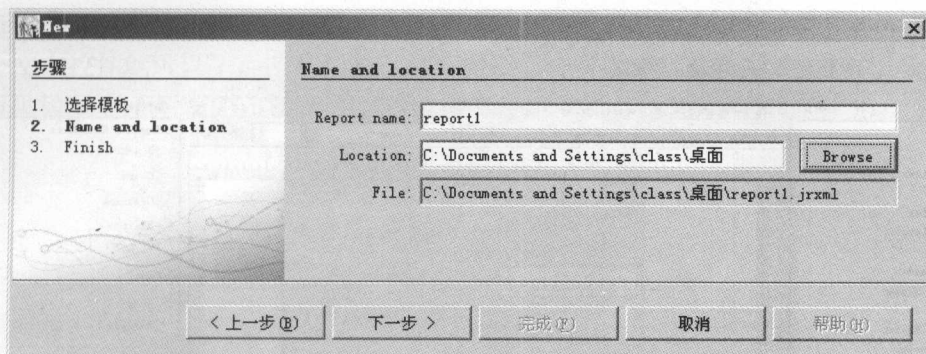


图 1.6 设置保存模板文件的路径

模板文件的扩展名为 jrxml，单击“下一步”按钮出现成功配置界面，效果如图 1.7 所示。

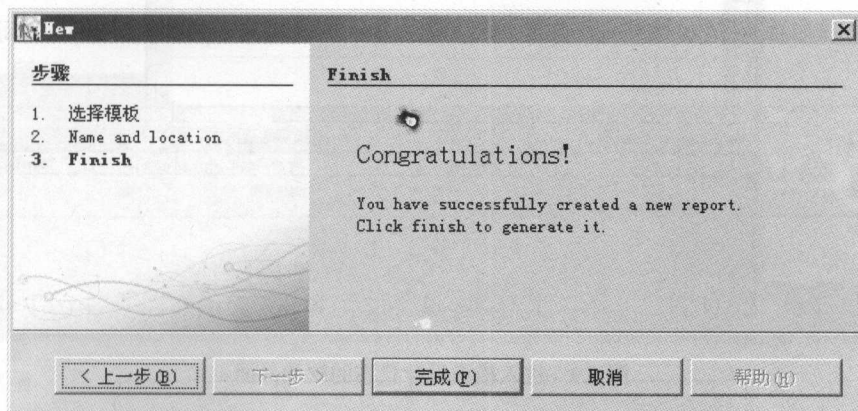


图 1.7 成功配置模板向导

单击“完成”按钮即可成功地在桌面上创建一个模板文件，效果如图 1.8 所示。

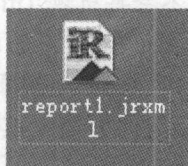


图 1.8 新建的模板文件

这时 iReport 的界面也发生了改变，可以在此界面上对模板文件进行处理，效果如图 1.9 所示。

报表框架被垂直分成若干个部分，每一个部分是一个 Band。每一个 Band 对象都有自己的特点，在生成报表的时候有些 Band 会打印一次，有些 Band 会打印多次。

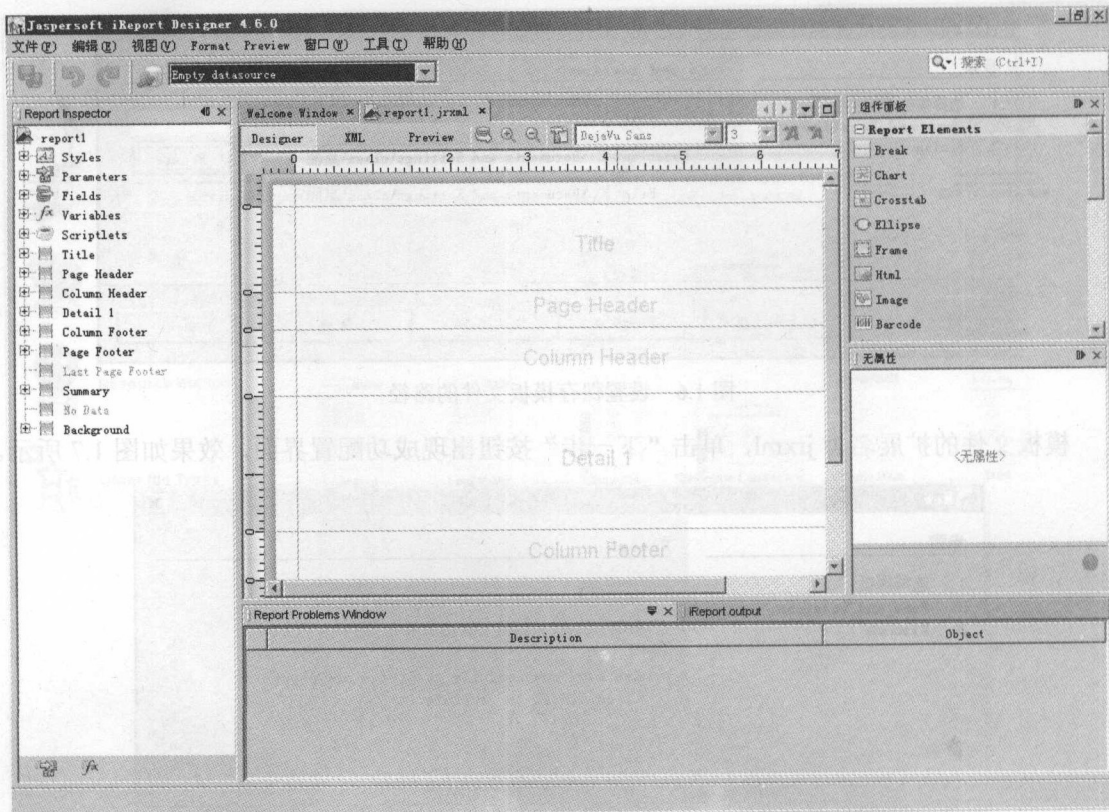


图 1.9 进入模板设计状态的软件界面

对这些 Band 的解释如下。

- Title (标题): Title Band 只在整个报表的第 1 页的最上面部分显示, 除了第 1 页以外, 不管报表中共有多少个页面也不会再出现 Title Band 中的内容。
- Page Header (页头): Page Header Band 中的内容将会在整个报表中的每一个页面中出现, 显示的位置在页面的上部, 如果是报表的第 1 页, Page Header 中的内容将显示在 Title Band 的下面, 除了第 1 页以外的其他所有页面中 Page Header 中的内容将显示在页面的最上端。
- Page Footer (页脚): 显示在当前页面的最下端。
- Detail 1 (详细): 报表内容段, 在这个 Band 中将要显示重复出现的内容, Detail 中的内容每页都会出现。
- Column Header (列头): Detail 1 Band 打印的是一张表, 而 Column Header Band 就是表中列的列头。
- Column Footer (列脚): Detail 1 Band 打印的是一张表, 而 Column Footer Band 就是表中列的列脚。
- Summary (统计): 表格的合计段, 出现在整个报表的最后一页中, 在 Detail 1 栏的后面, 一般用来统计报表中某一个或某几个字段的合计值。

本示例主要实现的功能就是打印一个静态的文本,所以就像使用.NET中的 WinForm 一样,找到那个文本控件就可以了,在右侧的“组件面板”中可以找到显示静态文本的 Static Text 控件,效果如图 1.10 所示。



图 1.10 显示静态文本的 Static Text 控件

把它拖动到 Detail 1 Band 中,并且设置它的 Text 属性为“Hello World-你好 世界”,效果如图 1.11 所示。

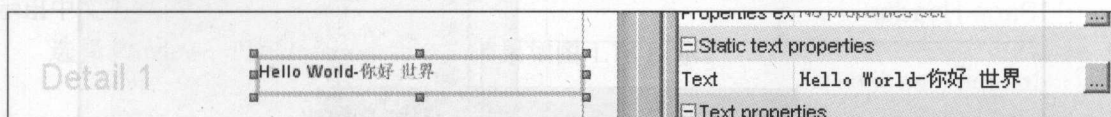


图 1.11 设置文本属性

将 Static Text 控件的宽高加大,可以发现文本默认是显示在控件的左上角,效果如图 1.12 所示。

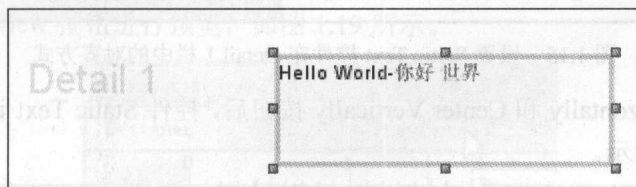


图 1.12 默认左上角显示

可以设置属性,将文本进行水平和垂直居中对齐,效果如图 1.13 所示。

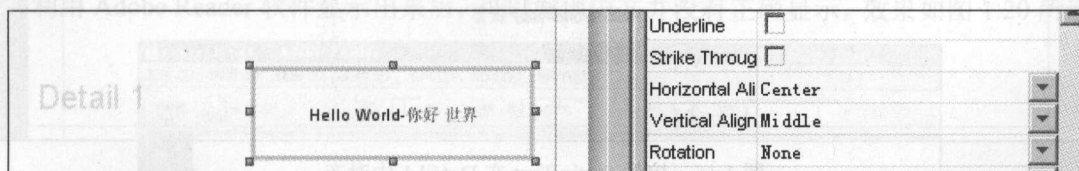


图 1.13 文本垂直水平居中对齐

虽然文本在控件 Static Text 内垂直水平居中对齐了,但控件 Static Text 并没有在 Detail 1 栏垂直水平居中对齐,因此选中 Static Text 控件继续设置,选择“窗口”→Formatting Tools 命令,效果如图 1.14 所示。

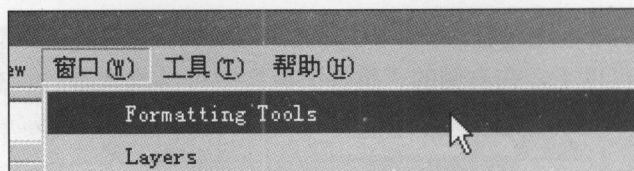


图 1.14 选择 Formatting Tools 命令

在面板中设置 Static Text 控件垂直和水平都居中对齐，效果如图 1.15 所示。

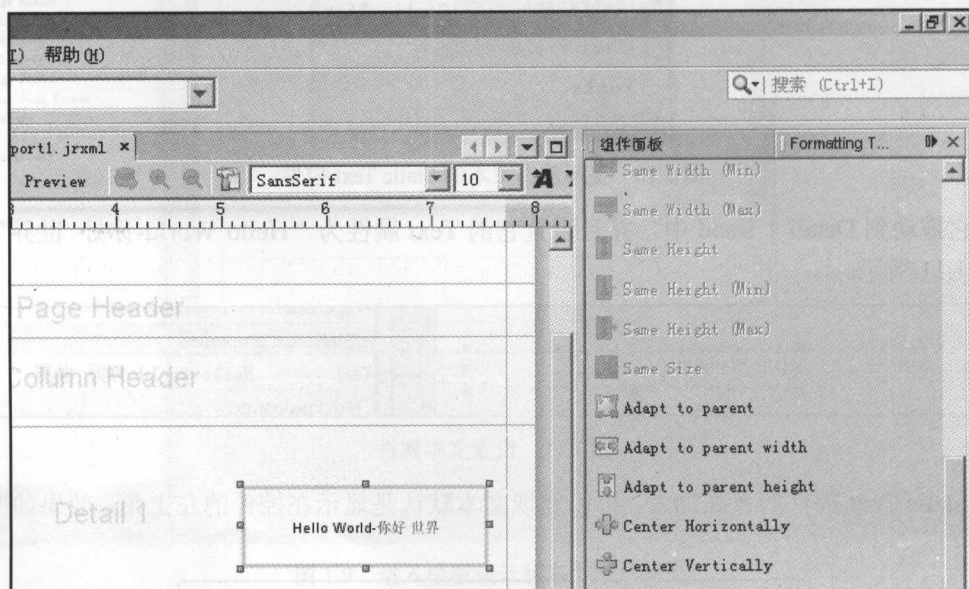


图 1.15 设置 Static Text 控件在 Detail 1 栏中的对齐方式

单击 Center Horizontally 和 Center Vertically 按钮后，控件 Static Text 设置了理想的对齐方式，效果如图 1.16 所示。

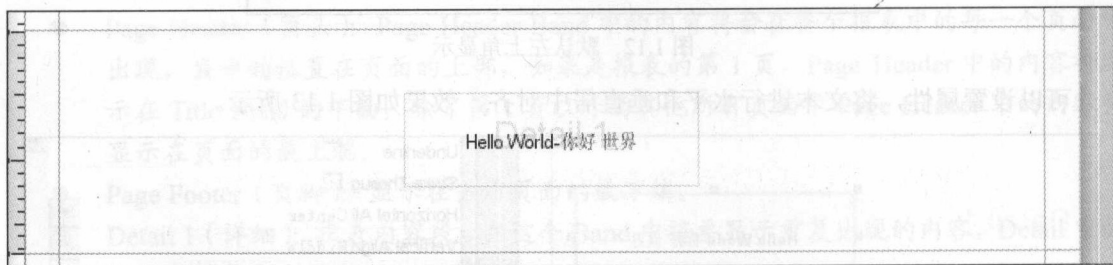



图 1.16 控件 Static Text 在 Detail 1 中对齐

设置完成后，单击左上角的“保存”按钮  保存这个模板文件。

至此报表的模板文件就创建完成了。

1.2.2 在 Web 项目中以 PDF 文件显示出来

虽然模板文件 report1.jrxml 创建结束了，但此时的模板文件并不能使用，必须编译成

report1.jasper 文件才可以，所以单击 Preview 预览按钮生成.jasper 文件，预览的效果如图 1.17 所示。

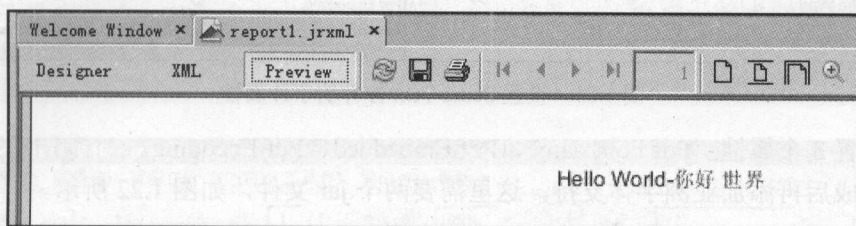



图 1.17 预览的效果

然后在桌面上即可创建出  文件。

虽然创建出了.jasper，但还是有一个非常重要的遗留问题，即：刚才是在 iReport 中进行预览，并且正确显示出中文，那如果用 Adobe Reader 软件预览 PDF 文件呢，是否还能正确显示出中文呢？

选择 Preview→PDF Preview 命令，效果如图 1.18 所示。

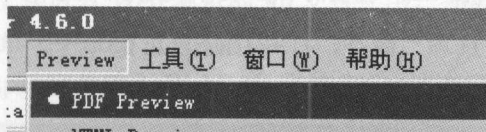


图 1.18 用 PDF 软件预览

然后再单击 Preview 按钮进行预览，如图 1.19 所示。

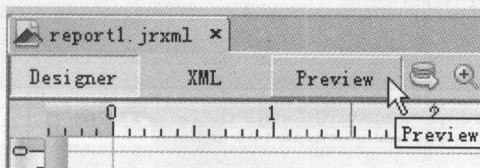


图 1.19 重新进行预览

利用 Adobe Reader 软件显示出来后，可以发现中文并没有正确显示，效果如图 1.20 所示。

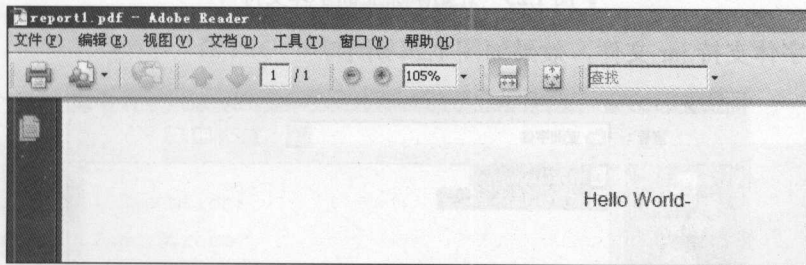


图 1.20 中文并没有正确显示

那如何解决呢？很简单，让 iReport 软件支持亚洲字体就可以了。首先是设置 Static Text 控件的字体属性，效果如图 1.21 所示。

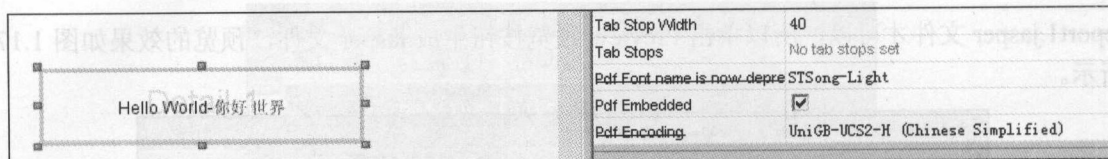


图 1.21 设置 Static Text 控件的字体属性

需要设置 3 个属性: Pdf Font name、Pdf Embedded、Pdf Encoding。
设置完成后再添加亚洲字体支持, 这里需要两个 jar 文件, 如图 1.22 所示。

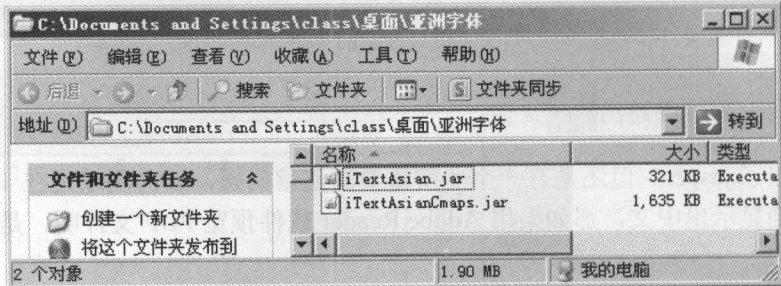


图 1.22 需要两个 jar 文件来支持中文字体

然后选择“工具”→“选项”命令, 打开 Classpath 选项卡, 单击 Add JAR 按钮添加亚洲字体 jar 文件的支持, 效果如图 1.23 所示。

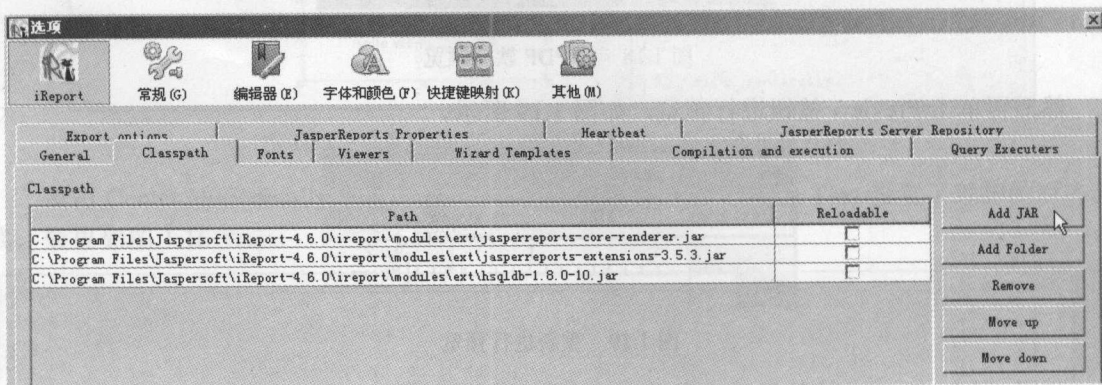


图 1.23 开始添加亚洲字体支持

选择两个字体支持 jar 文件, 效果如图 1.24 所示。

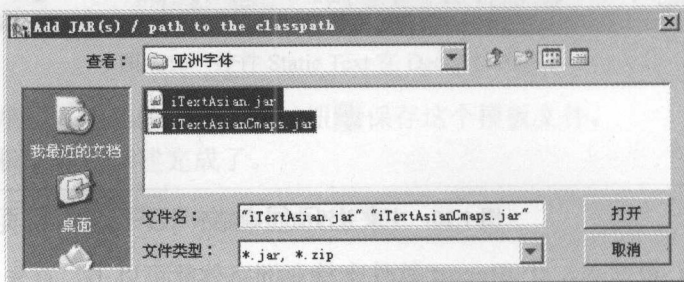


图 1.24 选择两个 jar 文件

单击“确定”按钮应用设置。



提示

设置完毕后一定要重启 iReport 软件，以加载最新的配置，再重新运行 PDF 即可正确显示出中文，效果如图 1.25 所示。

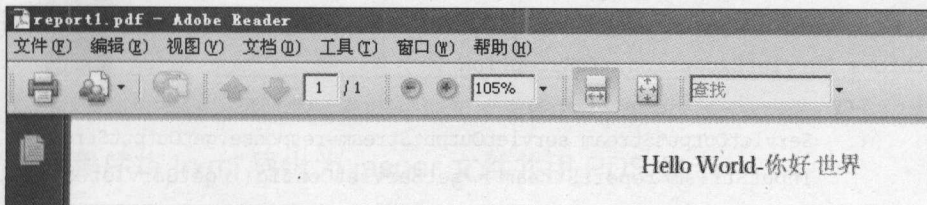


图 1.25 正确显示中文

以上步骤说明在 PDF 文件中已正确显示了中文字体，下一步就是创建一个 Web 项目，然后引入这个 .jasper 文件，将生成的 PDF 在 IE 上显示出来就可以了。

创建 Web 项目，把 jasperreports-4.6.0-project.zip 解压后将 jasperreports-4.6.0/dist/jasperreports-4.6.0.jar 中的文件放入 Web 项目的 lib 文件夹中，还要把 jasperreports-4.6.0/lib 中的所有 .jar 文件添加到 Web 项目的 lib 文件夹中。

要把桌面的 report1.jasper 文件放入 WebRoot 文件夹中，这点需要注意，项目结构如图 1.26 所示。

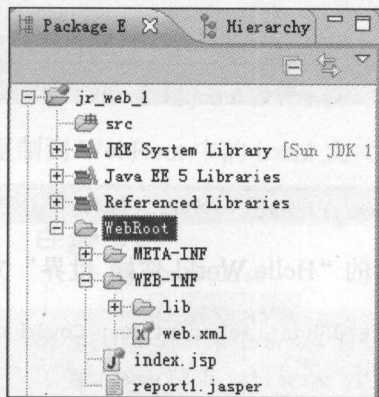


图 1.26 添加.jar 文件及导入.jasper 文件

创建一个路径为 test 的 Servlet，代码如下：

```
package controller;
import java.io.IOException;
import java.io.InputStream;
import java.util.HashMap;
import javax.servlet.ServletException;
import javax.servlet.ServletOutputStream;
import javax.servlet.http.HttpServlet;
```

```

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.jasperreports.engine.JREmptyDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JasperRunManager;
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream=response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report1.jasper");
            JasperRunManager.runReportToPdfStream(reportStream,
                servletOutputStream,new HashMap(), new JREmptyDataSource());
            response.setContentType("application/pdf");
            servletOutputStream.flush();
            servletOutputStream.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

部署项目，输入 URL：

http://localhost:8081/jr_web_1/test

页面上并没有出现应该显示的“Hello World-你好 世界”文本，而是出现如下异常：

```

net.sf.jasperreports.engine.JRRuntimeException: Could not load the following font :
pdfFontName : STSong-Light
pdfEncoding : UniGB-UCS2-H
isPdfEmbedded : true

```

为什么呢？因为缺少亚洲字体.jar 包文件，将 iTextAsian.jar 和 iTextAsianCmaps.jar 文件复制到 Web 项目的 lib 文件夹下，重启 Tomcat 并刷新 IE，显示出正确的中文效果，如图 1.27 所示。

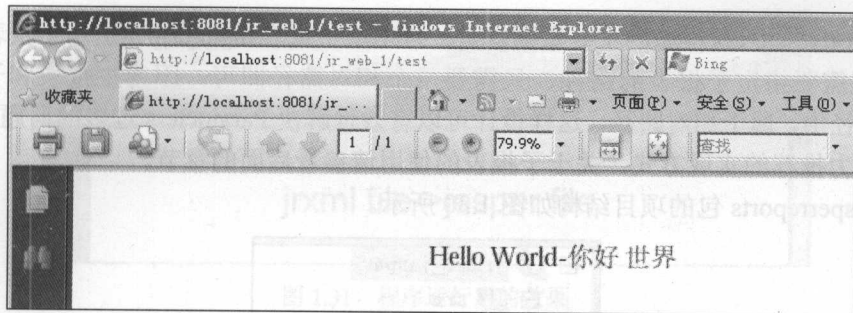


图 1.27 在 IE 中的 PDF 文件中显示出中文

1.2.3 利用程序将.jrxml 导出为.jasper 文件并用 PDF 显示

前面的章节是直接在 Web 项目中使用.jasper 文件, 其实在软件项目开发时, .jrxml 文件经常修改, 而且.jrxml 还要在项目中保留备份, 所以较为流行的做法是在项目中使用.jrxml 来生成.jasper 文件, 并用 PDF 文件显示出来, 想要实现这样的功能请继续后面的学习。

设计的报表模板样式, 如图 1.28 所示。

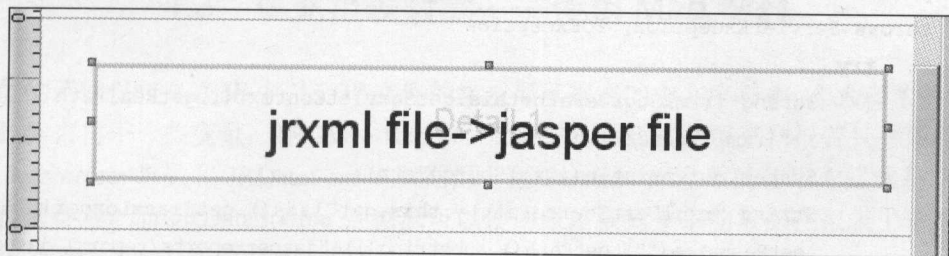


图 1.28 创建.jrxml 报表模板

创建 Web 项目, 将.jrxml 复制到 WebRoot 中的 jrxml 文件夹中, 项目结构如图 1.29 所示。

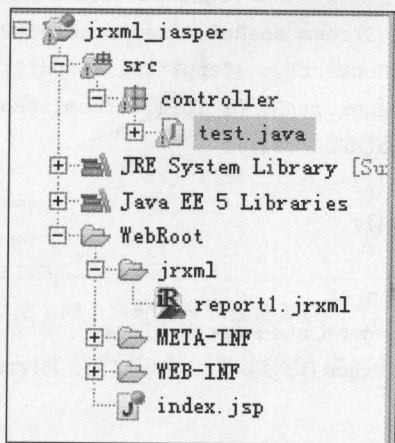


图 1.29 jrxml 文件夹中的.jrxml 文件

在此项目中的 src 路径下创建一个文件夹, 其实也是一个名称为 jasperreports 的包 (package), 创建它的主要作用就是 WebRoot 中的.jrxml 文件编译成.jasper 文件后要放入

src/jasperreports 包中,这样做的主要原因就是一旦出现主报表 main.jasper 引用子报表 sub.jasper 的时候,不需要指定子报表的路径,只需要一个相对路径即可,因为主报表和子报表都在 classpath 路径中,属于相对路径,这样设计可以有效地解决 Parameters 过多的问题,此方法也是本教程着力推荐的实现方式,关于子报表的使用请参看后面的章节。

加入 jasperreports 包的项目结构如图 1.30 所示。

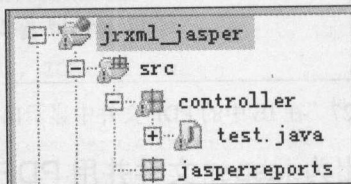


图 1.30 加入 jasperreports 包的项目结构

创建一个 Servlet, 核心代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            String jrxmlSourcePath=this.getServletContext().getRealPath ("/")
            + "jrxml\\report1.jrxml";
            System.out.println(jrxmlSourcePath);
            String jrxmlDestSourcePath = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1)+"jasperreports/report1.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePath,
                jrxmlDestSourcePath);
            InputStream isRef = new FileInputStream(new File(jrxmlDestSourcePath));
            ServletOutputStream sosRef = response.getOutputStream();
            response.setContentType("application/pdf");
            JasperRunManager.runReportToPdfStream(isRef,sosRef,new HashMap(),
                new JREmptyDataSource());
            sosRef.flush();
            sosRef.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

程序运行后的效果如图 1.31 所示。

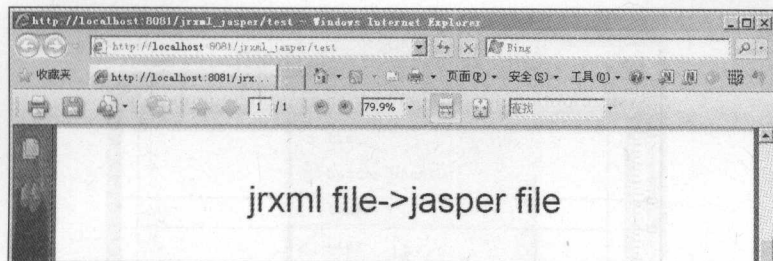


图 1.31 程序运行后的效果

编译成功后，在 Tomcat 软件中的指定路径下生成了 jasper 文件，如图 1.32 所示。

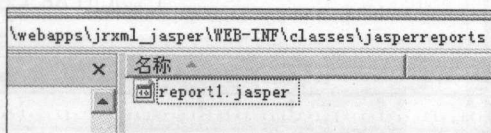


图 1.32 编译成功后的 jasper 文件

1.3 填充报表数据——使用 Map 参数

前面的示例仅仅是在 IE 中以 PDF 文件的格式显示静态的中文字符串，在大多数的情况下，打印的数据来自于一些变量，在 JasperReports 工具中传递数据并填充到报表只有两种方式，即使用 Parameters 参数和 JRDataSource 数据源，而 Parameters 参数的使用方式就是从 Servlet 向报表传递参数，而 JRDataSource 数据源可以使用原始数据表中的数据在报表上进行显示，如图 1.33 所示。

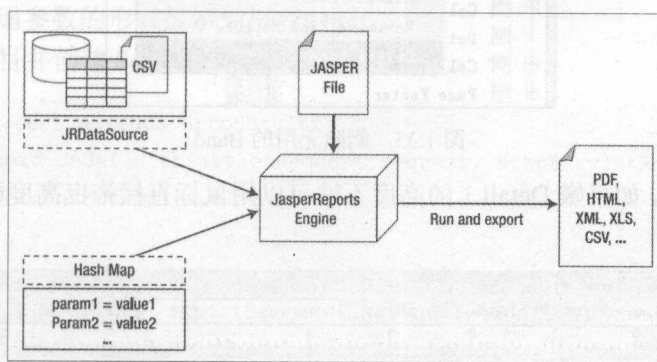


图 1.33 两种对报表传递参数的方式

这两种方式都可以通过 Servlet 进行传递，本示例就来实现通过 Parameters 传递数据到报表的功能。

1.3.1 新建报表模板文件

在 iReport 工具中新建一个报表，界面如图 1.34 所示。

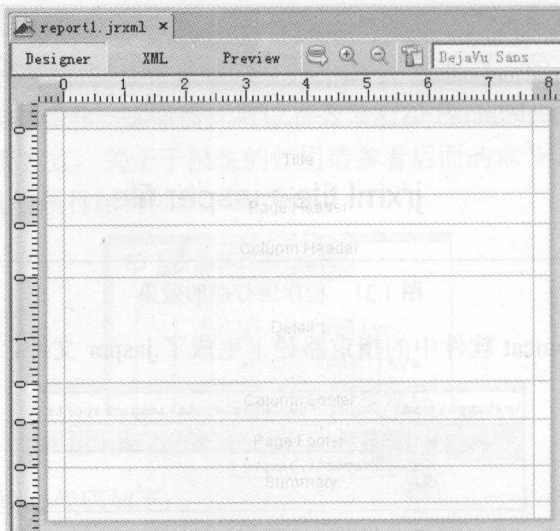


图 1.34 新建报表

在图 1.34 中有很多的 Band，如 Title、Page Footer 等。

本示例其实并不需要这么多的 Band，所以可以删除一些没有使用到的 Band，删除的办法是：在左边的 Report Inspector 面板中选择没有用到的 Band，单击鼠标右键，在弹出的快捷菜单中选择 Delete Band 命令即可，如图 1.35 所示。

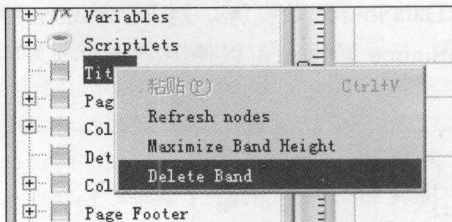


图 1.35 删除无用的 Band

只留有 Detail 1，如果嫌 Detail 1 的高度不够可以用鼠标直接拖曳高度即可，效果如图 1.36 所示。

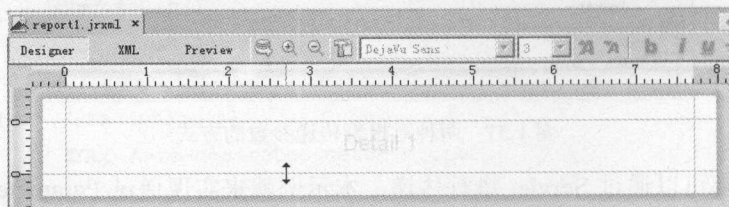


图 1.36 拖曳 Band 的高度

然后选择“组件面板”中的 Line 工具绘画表格，如图 1.37 所示。



图 1.37 使用 Line 工具

设计的表格样式如图 1.38 所示。

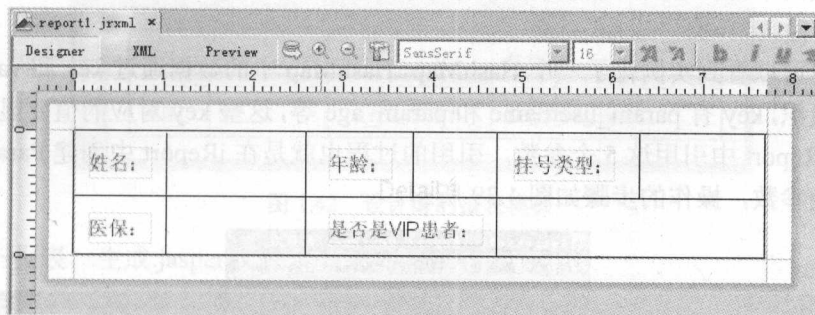


图 1.38 利用 Line 设计的表格

1.3.2 创建传递参数的 Servlet 对象

从图 1.38（利用 Line 设计的表格）中可以看到有些单元格的内容为空，这些内容就是通过在 Servlet 中传递参数的形式显示到报表中的。

新建 Web 项目并创建 Servlet，核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext().getResourceAsStream("report1.jasper");
            HashMap paramHashMap = new HashMap();
            paramHashMap.put("param_username", "高洪岩");
            paramHashMap.put("param_age", "40");
            paramHashMap.put("param_type", "专家号");
            paramHashMap.put("param_isHisSec", "是");
            paramHashMap.put("param_isVip", "是");
            JasperRunManager.runReportToPdfStream(reportStream,
            servletOutputStream, paramHashMap, new JREmptyDataSource());
            response.setContentType("application/pdf");
        }
    }
}
```



```

        servletOutputStream.flush();
        servletOutputStream.close();
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

从以上代码中可以看到，直接使用.jasper 文件即可进行报表的显示，而不需要从.jrxml 编译成.jasper 文件。

从代码中可以看到实例化了一个 HashMap，HashMap 中的数据通过 key 和 value 键值对的形式来进行组织，key 有 param_username 和 param_age 等，这些 key 对应的值要显示到报表上，所以需要在 iReport 中引用这 5 个参数，引用的过程也就是在 iReport 中创建 Parameters 对象。

创建一个参数，操作的步骤如图 1.39 所示。

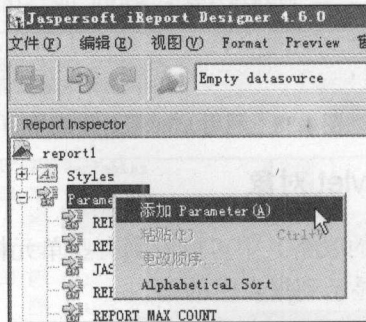


图 1.39 添加一个参数

对参数 Parameter 设置名称为 param_username，需要注意的是，此参数名称必须和 HashMap 中的 key 值相同，这样就可以从 Servlet 的 HashMap 中根据 key 值取到对应的值并且打印到报表中。

还要在属性面板中将属性值的提示功能去掉，如图 1.40 所示。

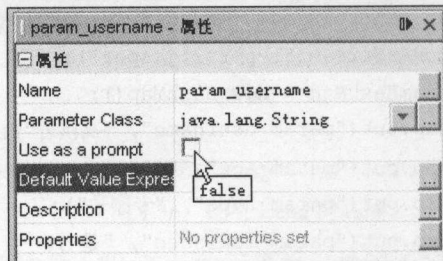


图 1.40 去掉 Use as a prompt 的勾选

创建好 5 个 Parameter，效果如图 1.41 所示。

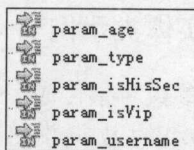


图 1.41 添加 5 个参数

把 5 个 Parameter 参数分别拖到空的单元格中, 然后设置它的字体大小及样式, 设置好的布局如图 1.42 所示。

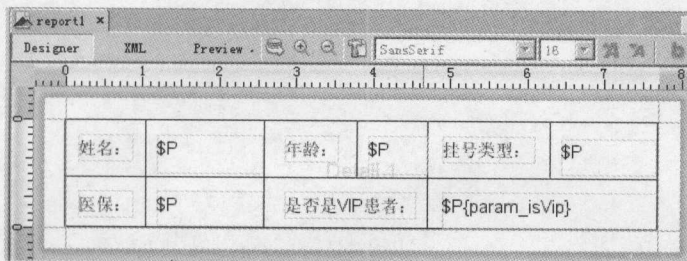


图 1.42 设计好的报表模板

预览一下报表, 生成.jasper 文件, 效果如图 1.43 所示。

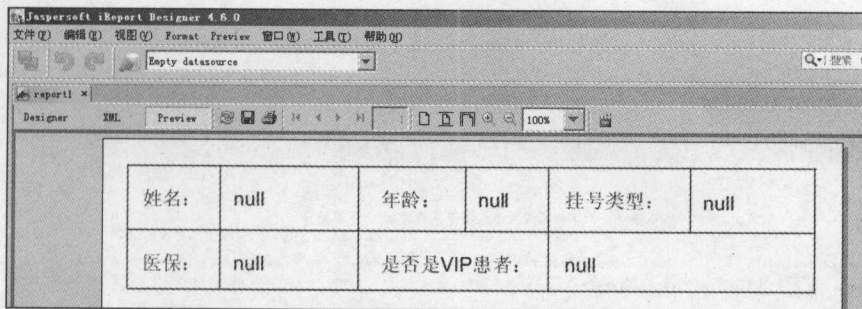


图 1.43 预览报表生成.jasper 文件

1.3.3 显示效果

把这个 report1.jasper 文件复制到 Web 项目的 WebRoot 文件夹下, 部署项目并输入 URL: <http://localhost:8081/reportHasStringEntity/test> 即可。在 IE 中以 PDF 文件的格式显示出了报表中的数据, 效果如图 1.44 所示。

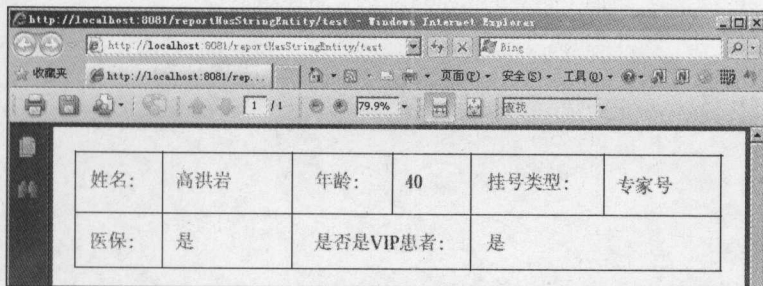


图 1.44 最终效果

1.3.4 打印 List 中 Userinfo.java 实体类示例

前面的内容演示了使用 Parameters 参数对象传递字符串的示例,本节将演示打印 List 接口中 Userinfo.java 实体类的示例,打印的数据源不是来自于 Parameters 对象,而是 JRBeanCollectionDataSource 对象。

新建 Web 项目,创建 Userinfo.java 实体类,结构如图 1.45 所示。

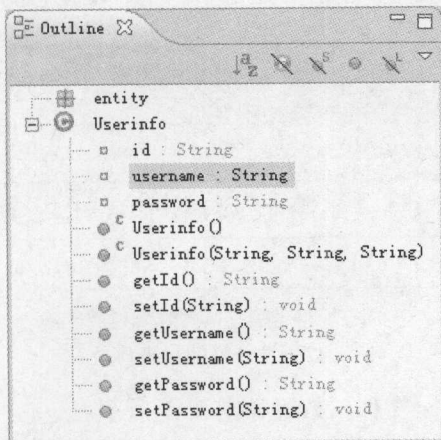


图 1.45 Userinfo.java 实体类结构

然后把 Userinfo.class 的路径配置到 iReport 软件中,如图 1.46 所示。

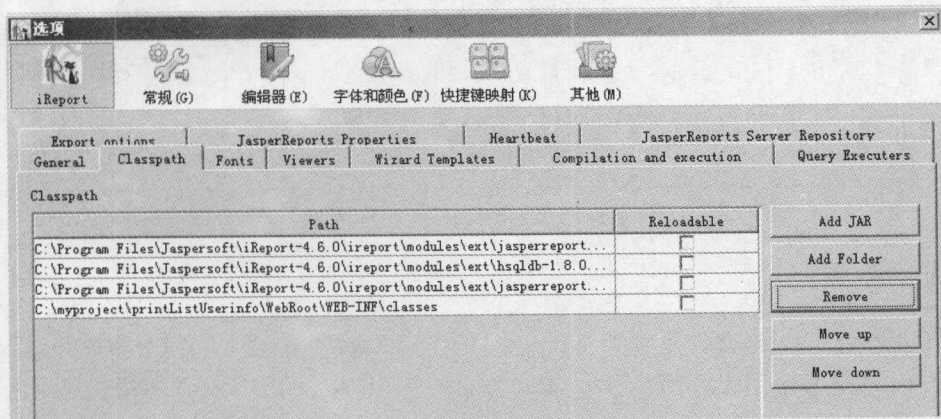



图 1.46 在 iReport 中配置.class 文件路径

单击  按钮添加数据源,选择 JavaBean,并且设置选项如图 1.47 所示。

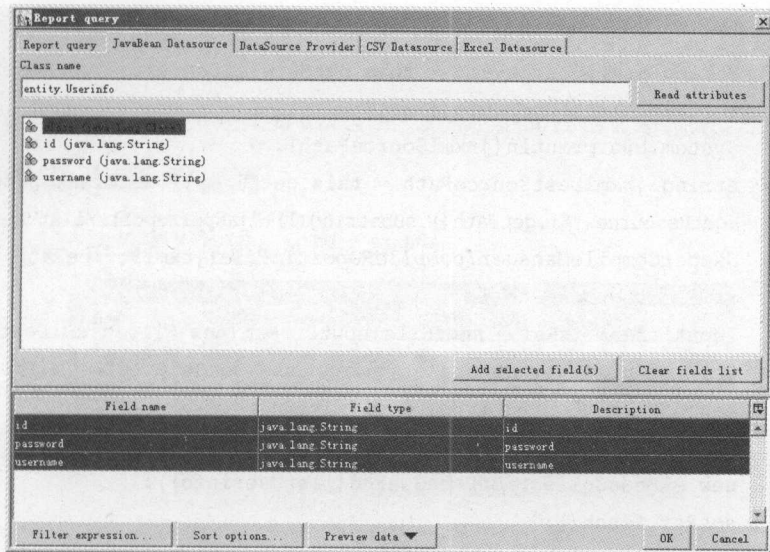


图 1.47 找到 UserInfo.class 并且读取里面的字段名称

单击 OK 按钮后 Fields 节点下出现 3 个字段名称, 如图 1.48 所示。



图 1.48 出现 3 个 Fields 对象

把这 3 个 Fields 对象添加到报表模板, 如图 1.49 所示。

id	password	username
\$F{id}	\$F{password}	\$F{username}

图 1.49 在报表模板中添加 3 个 Fields 对象

字段 id、username 及 password 的值来自于 List 中的 UserInfo.java 实体类, 而 List 就是打印报表的数据源, 创建 Servlet 核心代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List listUserInfo = new ArrayList();
            for (int i = 0; i < 100; i++) {
                listUserInfo.add(new UserInfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1)));
            }
        }
    }
}
```

```

    }
    String jrxmlSourcePath = this.getServletContext().getRealPath("/")
    + "jrxml\\listUserinfo.jrxml";
    System.out.println(jrxmlSourcePath);
    String jrxmlDestSourcePath = this.getClass().getClassLoader()
    .getResource("").getPath().substring(1)+"jasperreports/listUserinfo.jasper";
    JasperCompileManager.compileReportToFile(jrxmlSourcePath,
    rxmlDestSourcePath);
    InputStream isRef = new FileInputStream(new File(jrxmlDestSourcePath));
    ServletOutputStream sosRef = response.getOutputStream();
    response.setContentType("application/pdf");
    JasperRunManager.runReportToPdfStream(isRef, sosRef, new HashMap(),
    new JRBeanCollectionDataSource(listUserinfo));
    sosRef.flush();
    sosRef.close();
}
catch (JRException e)
{ // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

程序运行的效果如图 1.50 所示。

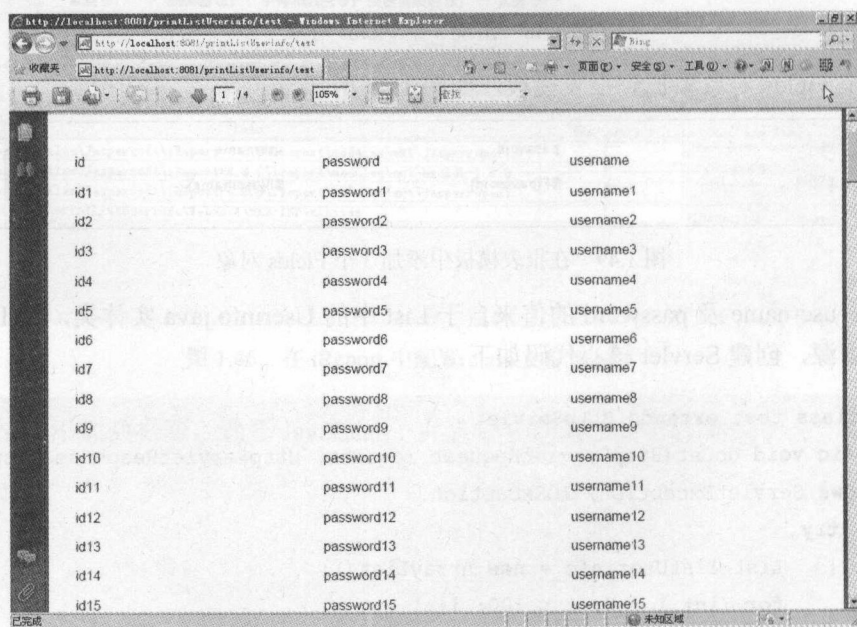


图 1.50 成功打印

1.4 填充报表数据——使用 JDBC 向导作为数据源

在 iReport 工具中可以直接连接数据表，把数据表中的数据显示出来。

1.4.1 新建报表 JDBC 数据源

单击“新建报表数据源”按钮，如图 1.51 所示。

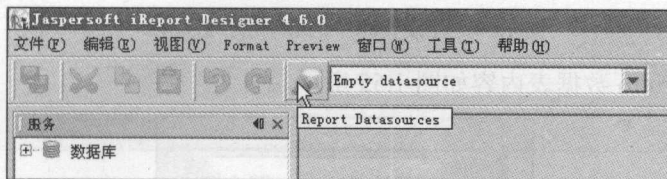


图 1.51 新建报表数据源

弹出一个设置数据源类型的对话框，效果如图 1.52 所示。

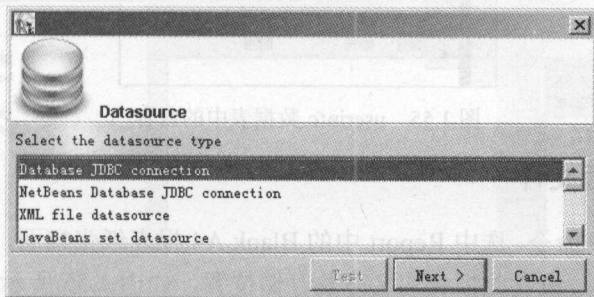


图 1.52 设置数据源类型

选择 Database JDBC connection 以 JDBC 的方式连接数据源，单击 Next 按钮后输入连接 MySQL 数据库的相关参数，并且单击 Test 按钮来测试是否正确连接数据库，效果如图 1.53 所示。

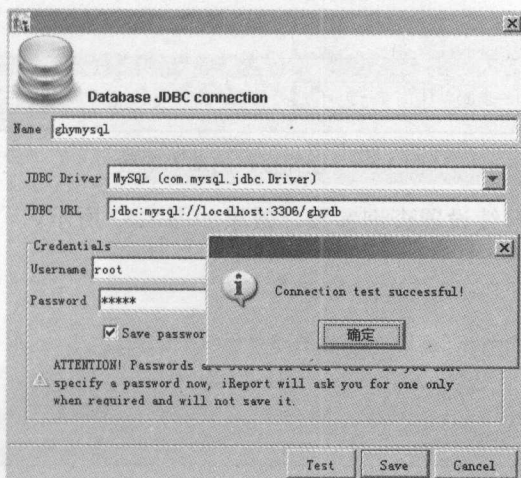


图 1.53 设置连接 MySQL 数据库的 JDBC 参数

连接成功后单击 **Save** 按钮保存这个报表数据源，并且在连接资源中列出此选项，效果如图 1.54 所示。

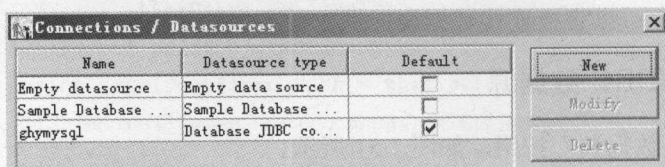


图 1.54 连接资源中的 MySQL 数据源

数据库中的 userinfo 数据表内容如图 1.55 所示。

id	username	password
1	a	aa
2	b	bb
3	c	cc
4	d	dd

图 1.55 userinfo 数据表中的内容

1.4.2 新建报表模板文件

选择“文件”→New 命令，选中 Report 中的 Blank A4 报表纸张幅面，然后单击 **Launch Report Wizard** 按钮启动向导创建报表，设置此报表的名称及存储位置，如图 1.56 所示。

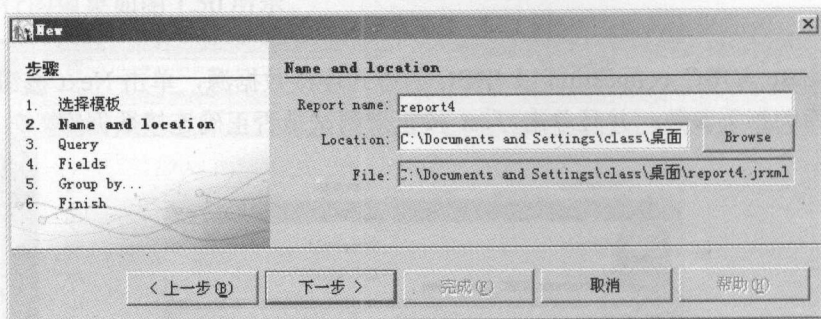


图 1.56 设置报表的基本参数

单击“下一步”按钮继续设置查询参数，效果如图 1.57 所示。

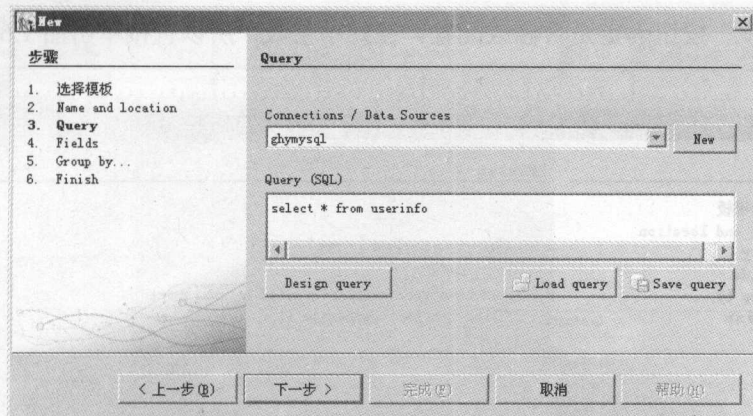


图 1.57 设置查询参数

单击“下一步”按钮将所有的列作为报表的显示列，效果如图 1.58 所示。

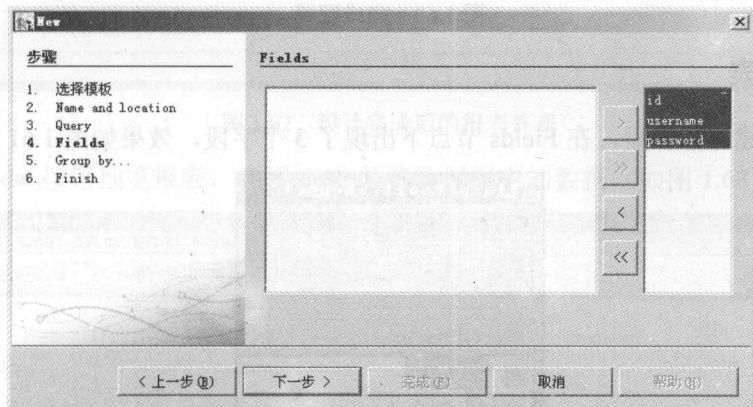


图 1.58 在报表上显示所有列

单击“下一步”按钮继续配置分组，由于本示例并不分组，所以选择默认值即可，效果如图 1.59 所示。

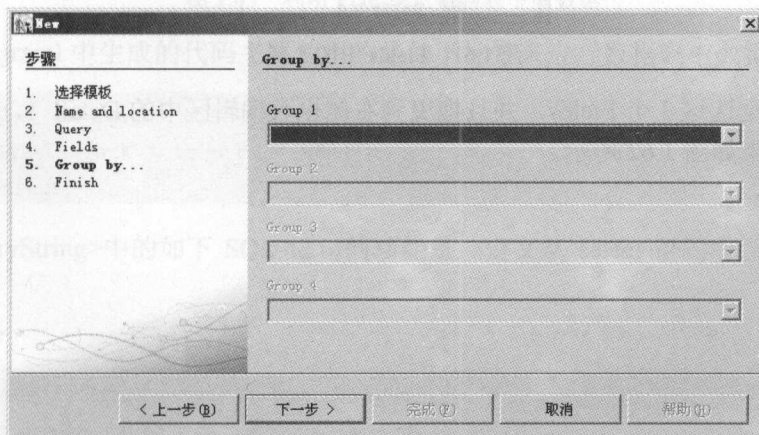


图 1.59 不设置分组

单击“下一步”按钮确定是否修改配置，在此不修改，所以直接单击图 1.60 中的“完成”按钮。

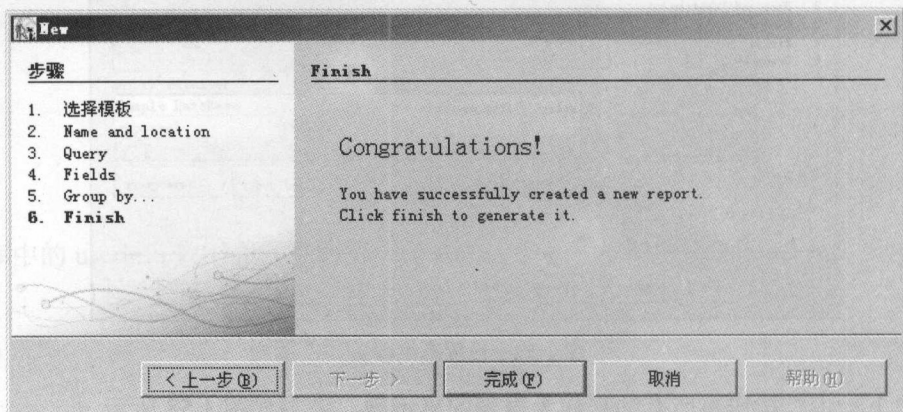


图 1.60 完成配置

1.4.3 设计报表

利用向导创建完报表后，在 Fields 节点下出现了 3 个字段，效果如图 1.61 所示。

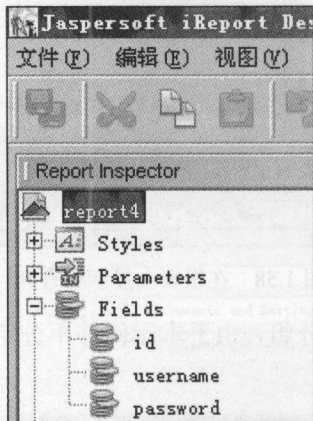


图 1.61 Fields 中的 3 列

利用 Ctrl 键全选这 3 个 Fields，并且拖曳到右侧报表编辑区中的 Detail 1，设置位置、大小等属性后，效果如图 1.62 所示。

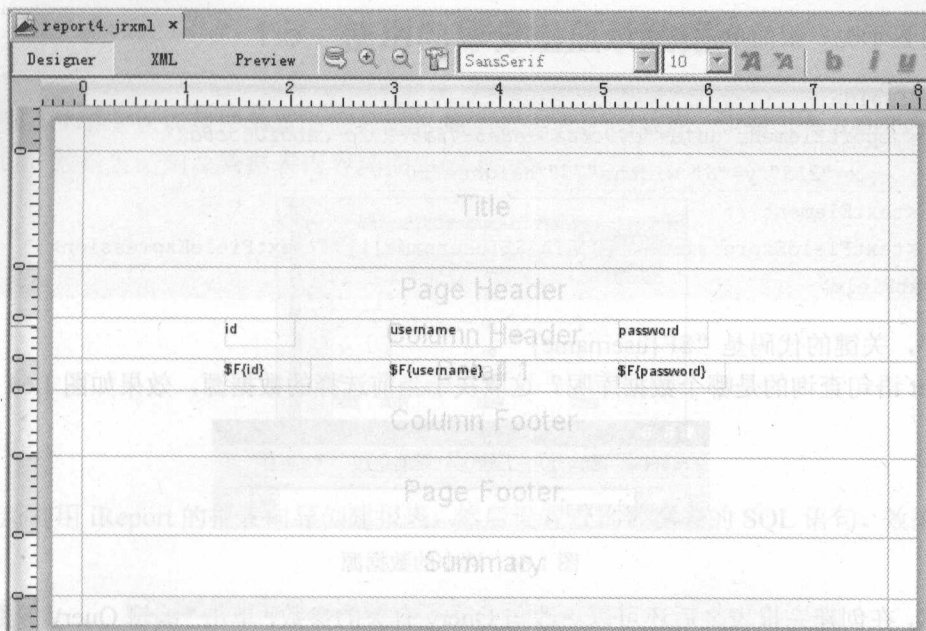


图 1.62 设计完成后的报表外观

单击 Preview 按钮预览报表，利用 PDF 预览数据表中的数据，如图 1.63 所示。

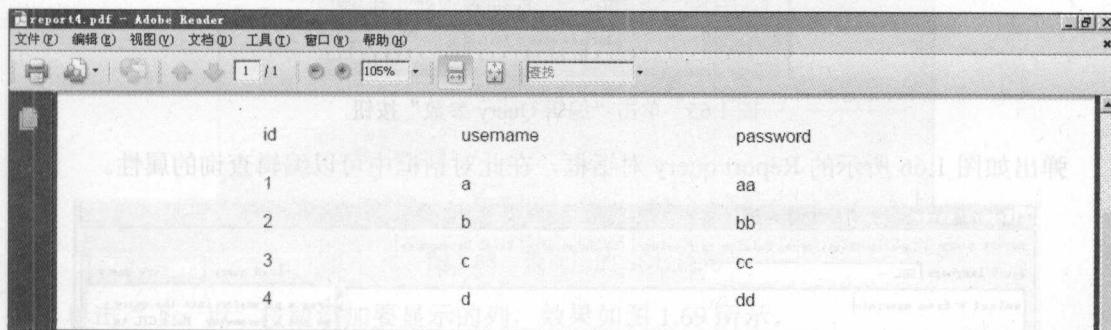


图 1.63 利用 PDF 预览数据表中的数据

报表文件.jrxml 中生成的代码主要使用的是如下标签来进行数据表中数据的显示：

```
<queryString language="SQL">
    <![CDATA[select * from userinfo]]>
</queryString>
```

标签<queryString>中的如下 SQL 语句的功能是：定义从 select 语句中返回哪些字段显示到报表上。

代码如下：

```
<field name="id" class="java.lang.Integer" />
<field name="username" class="java.lang.String" />
<field name="password" class="java.lang.String" />
```

如何显示呢？很简单，使用如下代码即可：

```
<textField>
  <reportElement uuid="b49c6aa5-e8e6-46a8-952c-cab5f0c5c9d2"
    x="213" y="0" width="77" height="20" />
  <textElement />
  <textFieldExpression><![CDATA[${username}]]></textFieldExpression>
</textField>
```

其中，关键的代码是“``${username}``”。

`select` 语句查询的是哪个数据库呢？这取决于当前选择的数据源，效果如图 1.64 所示。

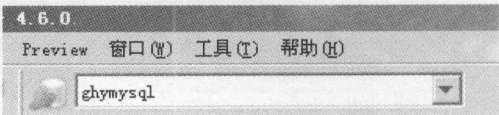


图 1.64 选中的数据源

当然，在创建完报表之后还可以更改与 Query 有关的参数，单击“编辑 Query 参数”按钮，如图 1.65 所示。

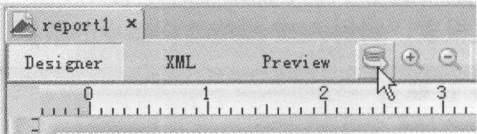


图 1.65 单击“编辑 Query 参数”按钮

弹出如图 1.66 所示的 Report query 对话框，在此对话框中可以编辑查询的属性。

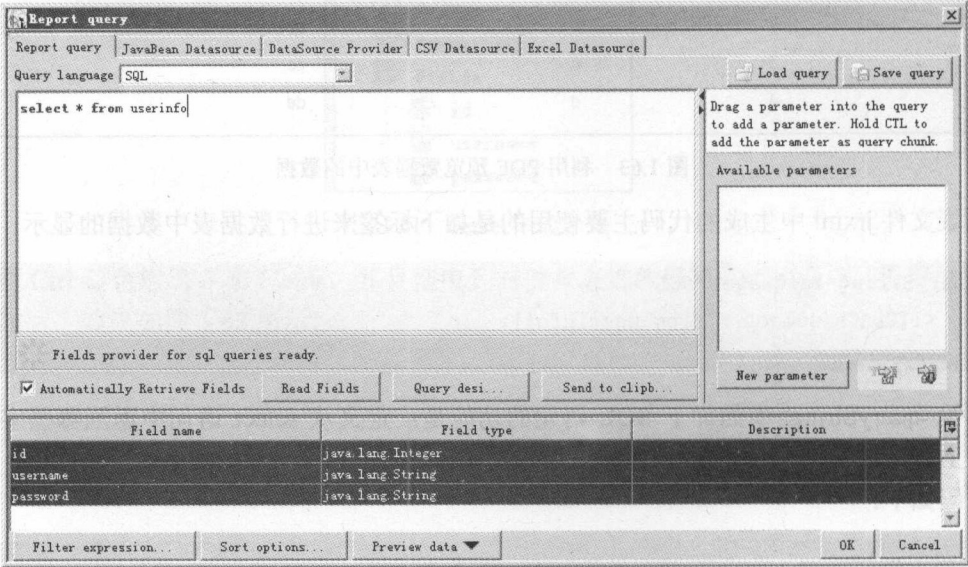
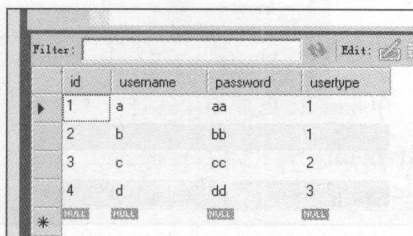


图 1.66 Report query 对话框

1.5 使用向导分组显示数据

在第 1.4 节中仅仅是用最普通的方式来显示数据表中的数据，那如果有分组的情况呢？更改 userinfo 数据表结构及数据表内容如图 1.67 所示。



	id	username	password	usertype
▶	1	a	aa	1
	2	b	bb	1
	3	c	cc	2
	4	d	dd	3

* NULL NULL NULL NULL

图 1.67 更改表结构及数据表内容后的结果

重新利用 iReport 的报表向导创建报表，然后设置查询数据表的 SQL 语句，效果如图 1.68 所示。

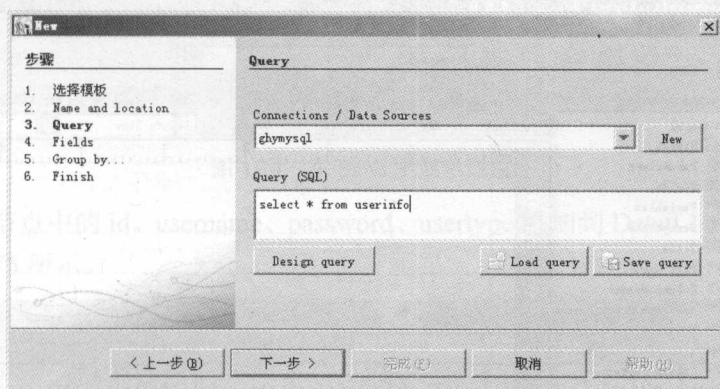


图 1.68 查询用的 SQL 语句

单击“下一步”按钮添加要显示的列，效果如图 1.69 所示。

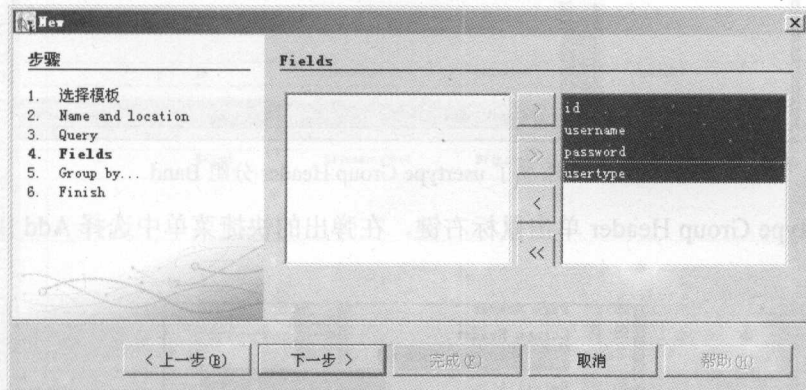


图 1.69 添加要显示的列

单击“下一步”按钮继续配置，一定要对 usertype 进行分组显示，效果如图 1.70 所示。

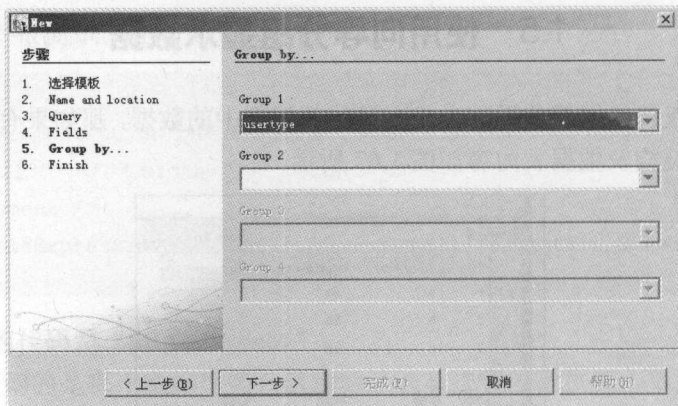


图 1.70 对 usertype 分组显示

单击“下一步”按钮完成配置。创建出的报表模板结构，即添加了 usertype Group Header 分组 Band 的效果如图 1.71 所示。

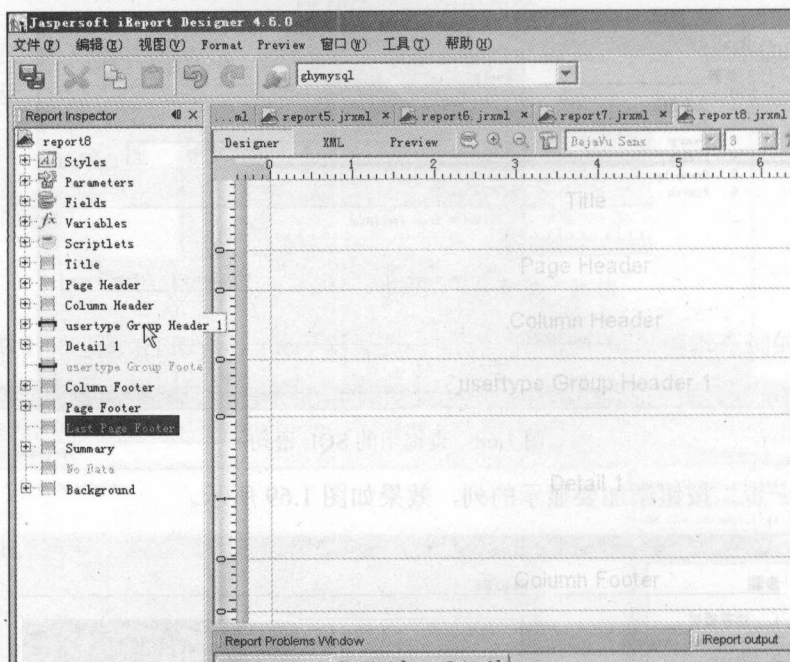


图 1.71 添加了 usertype Group Header 分组 Band

选中 usertype Group Header 单击鼠标右键，在弹出的快捷菜单中选择 Add Band 命令，如图 1.72 所示。

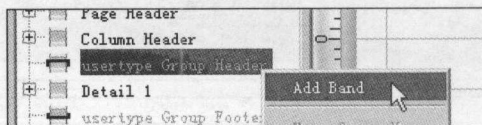


图 1.72 添加 Band

再把 Fields 节点下的 usertype 拖曳到这个 Band，效果如图 1.73 所示。

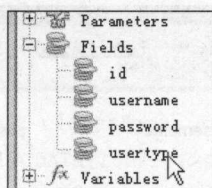


图 1.73 将 usertype 拖曳到 Band 中

弹出设置在该 Band 中显示什么内容的对话框，在本示例中设置 Band 只显示原值，效果如图 1.74 所示。

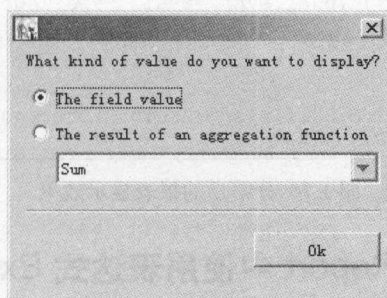


图 1.74 在 Band 中显示原值

再把 Fields 节点中的 id、username、password、usertype 添加到 Detail 1 中，设计完成后的报表模板如图 1.75 所示。

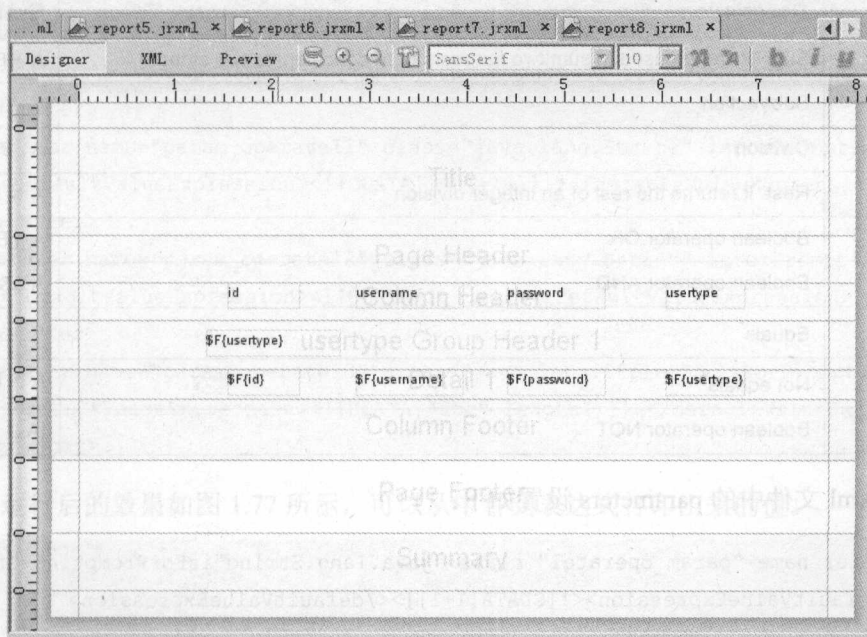
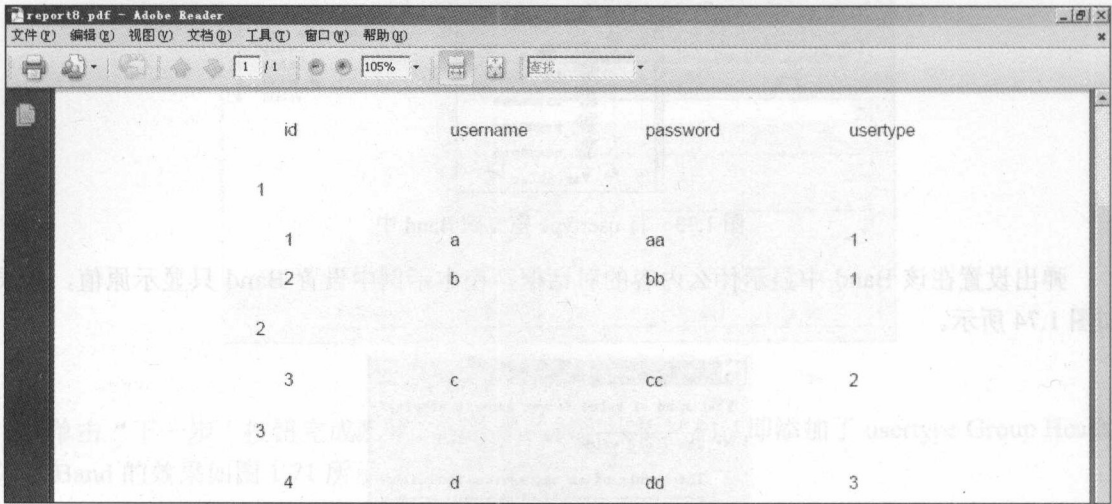


图 1.75 设计完成后的报表模板

单击 Preview 按钮预览报表，效果如图 1.76 所示。



id	username	password	usertype
1			
1	a	aa	1
2	b	bb	1
2			
3	c	cc	2
3			
4	d	dd	3

图 1.76 分组后的报表显示效果

1.6 在 iReport 中使用表达式 Expression

在 iReport 中还支持表达式的使用，iReport 中支持的表达式如表 1.1 所示。

表 1.1 iReport 支持的表达式

Operator	Description	Example
+	Sum (it can be used to sum two numbers or to concatenate two strings)	A + B
-	Subtraction	A - B
/	Division	A / B
%	Rest, it returns the rest of an integer division	A % B
	Boolean operator OR	A B
&&	Boolean operator AND	A && B
==	Equals*	A == B
!=	Not equals†	A != B
!	Boolean operator NOT	!A

报表.jrxml 文件中的 parameters 设置如下：

```
<parameter name="param_operate1" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[1+1]]></defaultValueExpression>
</parameter>
<parameter name="param_operate2" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[100-1]]></defaultValueExpression>
```

```

</parameter>
<parameter name="param_operate3" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[3*2]]></defaultValueExpression>
</parameter>
<parameter name="param_operate4" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[100/2]]></defaultValueExpression>
</parameter>
<parameter name="param_operate5" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[5%2]]></defaultValueExpression>
</parameter>
<parameter name="param_operate6" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[true || false]]></defaultValueExpression>
</parameter>
<parameter name="param_operate7" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[true && true]]></defaultValueExpression>
</parameter>
<parameter name="param_operate8" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[2==1]]></defaultValueExpression>
</parameter>
<parameter name="param_operate9" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA["高洪岩".equals("高洪岩")]]>
  </defaultValueExpression>
</parameter>
<parameter name="param_operate10" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[!"a".equals("a")]]></defaultValueExpression>
</parameter>
<parameter name="param_operate11" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[100!=101]]></defaultValueExpression>
</parameter>
<parameter name="param_operate12" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA[!true]]></defaultValueExpression>
</parameter>
<parameter name="param_operate13" class="java.lang.String" isForPrompting="false">
  <defaultValueExpression><![CDATA["abc".length()]]></defaultValueExpression>
</parameter>

```

程序运行后的效果如图 1.77 所示，可以从中看到表达式打印出来的值。

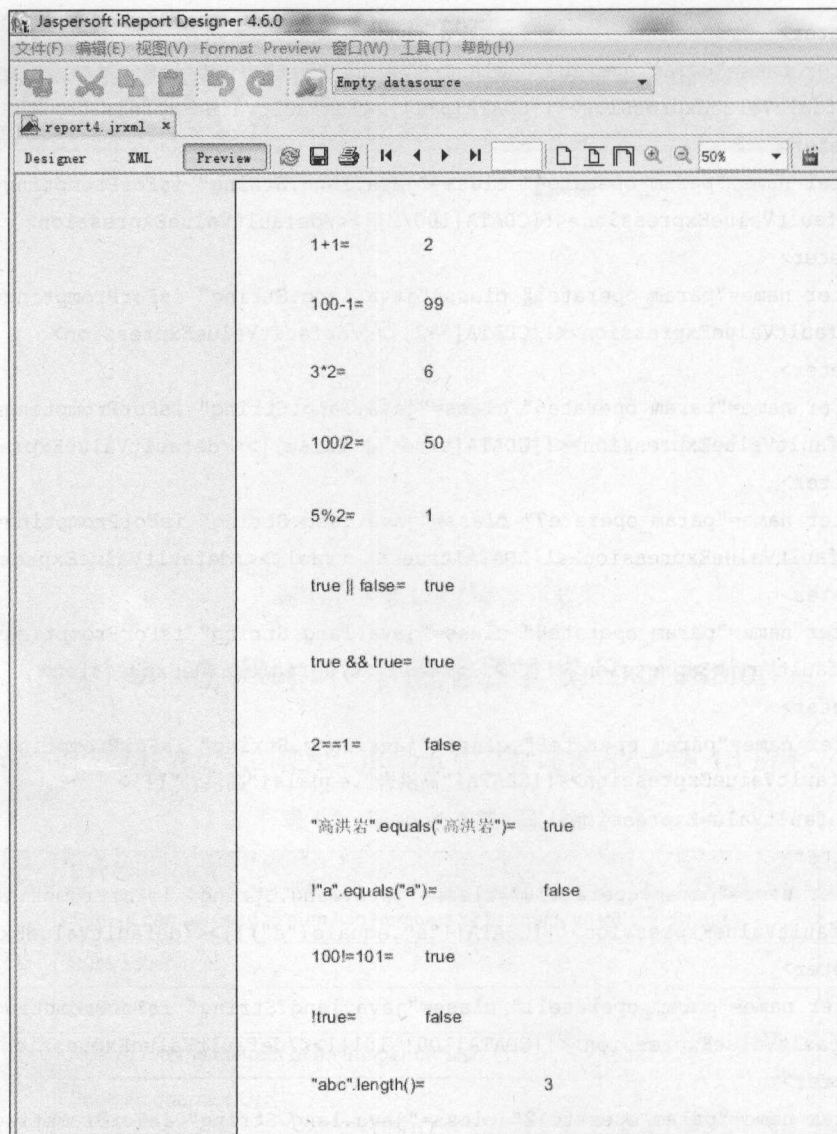


图 1.77 表达式打印出来的值

从图 1.77 中可以看到，还有调用对象的方法，这些方法的使用请参考帮助文档，这里不再赘述。

不过还是建议不要在 .jrxml 文件中编写过于复杂的表达式，这样会把业务的判断放入 iReport 报表层，将严重破坏 Model 层的封装业务，即不要把一件事情放在两个地方去做。

1.7 将报表导出为 PDF 文件

本示例要把 .jasper 文件导出到 PDF 文件，而不是像前面示例中使用 IE 显示 PDF 文件。在导出指定文件格式之前需要知道导出的一个大体过程，如图 1.78 所示。

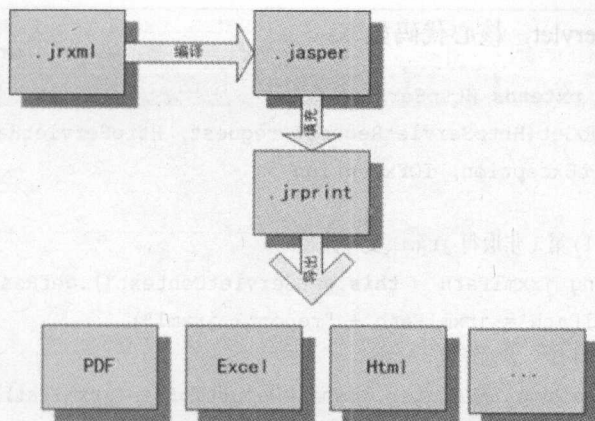


图 1.78 导出报表的大体过程

导出的大体过程如下。

- 步骤 01** 扩展名为 .jrxml 的文件为标准报表模板 XML 格式的文件，该文件定义了报表的格式和数据构成，可以使用 iReport 软件以可视化的方式生成并编辑 .jrxml 报表模板文件。
- 步骤 02** 将创建出来的 .jrxml 模板文件经过 JasperReports API 编译后生成扩展名为 .jasper 的二进制文件。
- 步骤 03** 可以调用 JasperReports API 针对 .jasper 文件进行数据和参数的填充，生成扩展名为 .jrprint 的文件。
- 步骤 04** 再调用 JasperReports API 将 .jrprint 文件最终导出成 PDF、Excel、HTML 等各种格式的文件。

下面用代码的方式来演示上面 4 个步骤，首先新建一个 .jrxml 模板，最简单的 .jrxml 文件如图 1.79 所示。

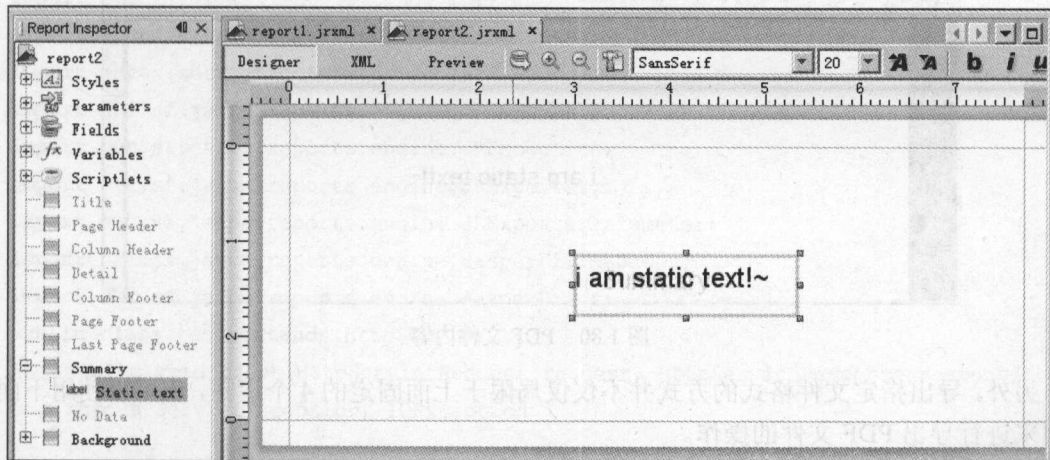


图 1.79 最简单的 .jrxml 文件

其次要创建一个 Servlet，核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            // (1) 第 1 步取得 jrxml 文件路径
            String jrxmlPath = this.getServletContext().getRealPath("/");
            jrxmlPath = jrxmlPath + "report2.jrxml";
            // (2) 第 2 步编译成 jasper
            JasperCompileManager.compileReportToFile(jrxmlPath, jrxmlPath
                + "report2.jasper");
            // (3) 第 3 步生成 JasperPrint 对象
            JasperPrint jasperPrint = JasperFillManager.fillReport(jrxmlPath
                + "report2.jasper", new HashMap(), new JREmptyDataSource());
            // (4) 第 4 步导出成 PDF 指定格式
            JasperExportManager.exportReportToPdfFile(jasperPrint, "c://demo.pdf");
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

运行 Servlet 后在 C 盘出现 demo.pdf 文件，双击该文件将显示如图 1.80 所示的内容。

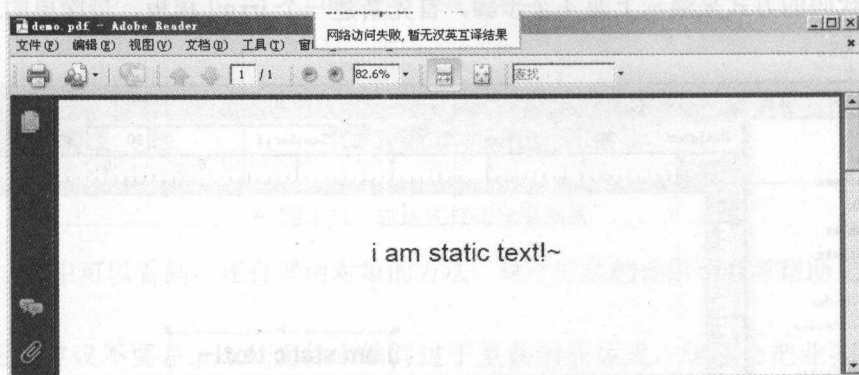


图 1.80 PDF 文件内容

另外，导出指定文件格式的方式并不仅仅局限于上面固定的 4 个步骤，还可以使用下面的示例来进行导出 PDF 文件的操作。

新建 Web 项目，并且把 report3.jasper 文件复制到 WebRoot 文件夹中，在复制.jasper 文件之前要先新建一个报表，设计具有中文的报表如图 1.81 所示。

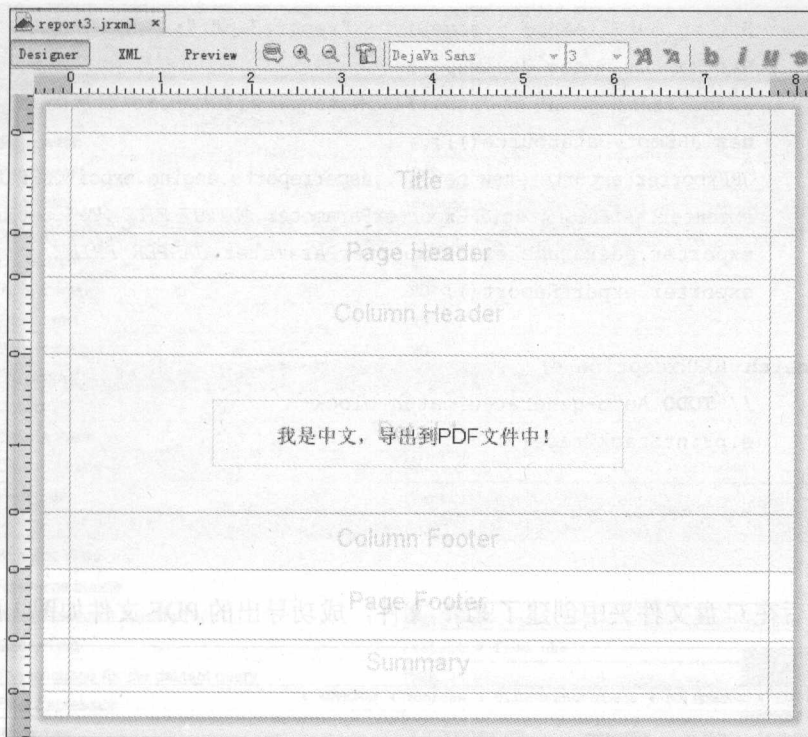


图 1.81 设计具有中文的报表

Servlet 代码如下:

```
package controller;
import java.io.IOException;
import java.util.HashMap;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import net.sf.jasperreports.engine.JREmptyDataSource;
import net.sf.jasperreports.engine.JRException;
import net.sf.jasperreports.engine.JRExporter;
import net.sf.jasperreports.engine.JRExporterParameter;
import net.sf.jasperreports.engine.JasperFillManager;
import net.sf.jasperreports.engine.JasperPrint;
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            String saveDiv = request.getRealPath("/");
            String fileName = saveDiv + "report3.jasper";
```



```

String outFileName = saveDiv + "report3.pdf";
HashMap hm = new HashMap();
JasperPrint print = JasperFillManager.fillReport(fileName, hm,
new JREmptyDataSource());
JRExporter exporter=new net.sf.jasperreports.engine.export.JRPdfExporter();
exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,outFileName);
exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
exporter.exportReport();
}
catch (JRException e)
{ // TODO Auto-generated catch block
e.printStackTrace();
}
}
}

```

程序运行后在 C 盘文件夹中创建了 PDF 文件，成功导出的 PDF 文件如图 1.82 所示。

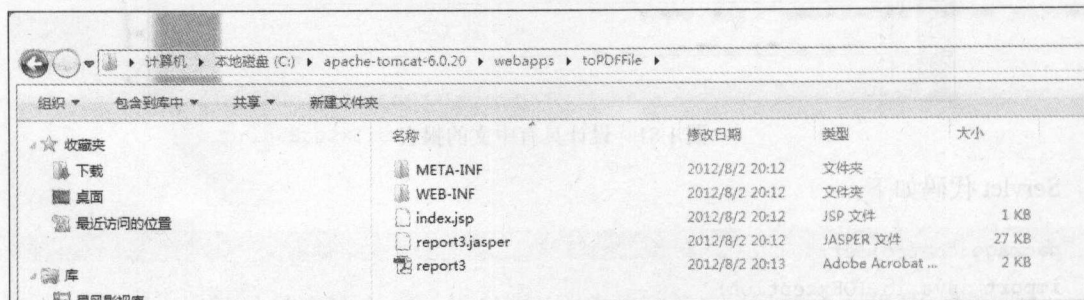


图 1.82 成功导出的 PDF 文件

双击 report3.pdf 文件后可以看到已经显示出了正确的中文，效果如图 1.83 所示。

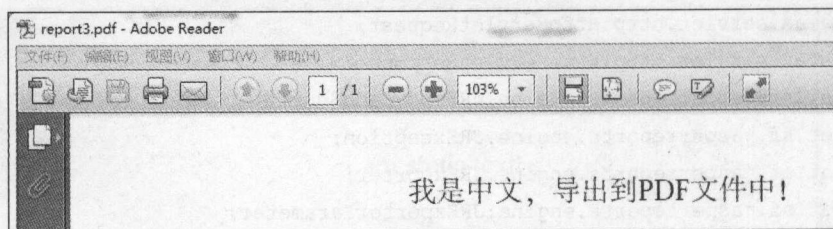


图 1.83 PDF 中的中文

1.8 报表的常用属性

可以在“属性”面板中设置一些有关报表的属性，效果如图 1.84 所示。

Report Inspector		report7 - 属性
Report name	report7	
Page size		
Page width	595	
Page height	842	
Orientation	Portrait	
Margins		
Left margin	20	
Right margin	20	
Top margin	20	
Bottom margin	20	
Columns		
Columns	1	
Column Width	555	
Column space	0	
Print order	Vertical	
More...		
Scriptlet class		
Resource bundle		
When Resource Missing Type	Type Null	
Query Text	select * from abc	
The language for the dataset query	SQL	
Filter Expression		
Properties	3 properties set	
Title on a new page	<input type="checkbox"/>	
Summary on a new page	<input type="checkbox"/>	
Summary with Page Header and Footer	<input type="checkbox"/>	
Float column footer	<input type="checkbox"/>	
Ignore pagination	<input type="checkbox"/>	
Column Direction	Left to Right	
When No Data	No Data Section	
Language	Groovy	
Format Factory Class		
Imports	No imports set	

图 1.84 报表的属性

可以在此面板中设置报表的宽度、高度及打印的方向，还可以设置报表中列的数目、外边距等。其中最为重要的还是分列、分栏的显示，在默认的情况下是不分栏的，那分栏的效果，也就是多列的效果如何呢？本节将给予介绍。

1.8.1 分栏分列的效果

设置报表的列数为 2，属性设置如图 1.85 所示。

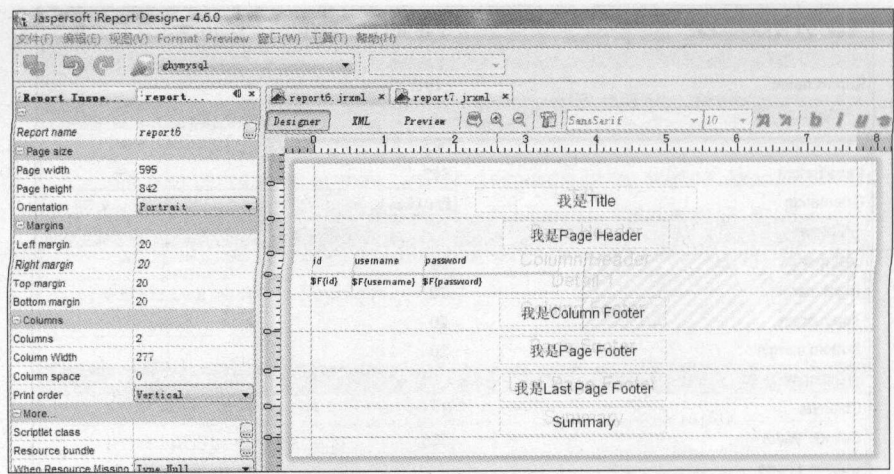


图 1.85 设置 2 列并且更改报表设置

双列的报表预览效果如图 1.86 所示。

我是Title					
我是Page Header					
id	username	password	id	username	password
1	a	aa	31	a	aa
2	b	bb	32	a	aa
3	c	cc	33	a	aa
4	d	dd	34	a	aa
5	e	ee	35	a	aa
6	a	aa	36	a	aa
7	a	aa	37	a	aa
8	a	aa	38	a	aa
9	a	aa	39	a	aa
10	a	aa	40	a	aa
11	a	aa	41	a	aa
12	a	aa	42	a	aa
13	a	aa	43	a	aa
14	a	aa	44	a	aa
15	a	aa	45	a	aa
16	a	aa	46	a	aa
17	a	aa	47	a	aa
18	a	aa	48	a	aa
19	a	aa	49	a	aa
20	a	aa	50	a	aa
21	a	aa	51	a	aa
22	a	aa	52	a	aa
23	a	aa	53	a	aa
24	a	aa	54	a	aa
25	a	aa	55	a	aa
26	a	aa	56	a	aa
27	a	aa	57	a	aa
28	a	aa	58	a	aa
29	a	aa	59	a	aa
30	a	aa	60	a	aa
我是Column Footer			我是Column		
我是Page Footer					

图 1.86 双列的报表效果

1.8.2 Title 和 Summary 在单独的页面打印

当然还可以将报表的 Title 和 Summary 设置在单独的页面打印, 属性设置如图 1.87 所示。

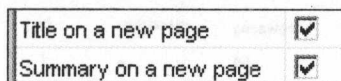


图 1.87 将 Title 和 Summary 在单独的页面打印

重新对 .jrxml 进行界面布局, 重新布局的 Title 示例如图 1.88 所示。

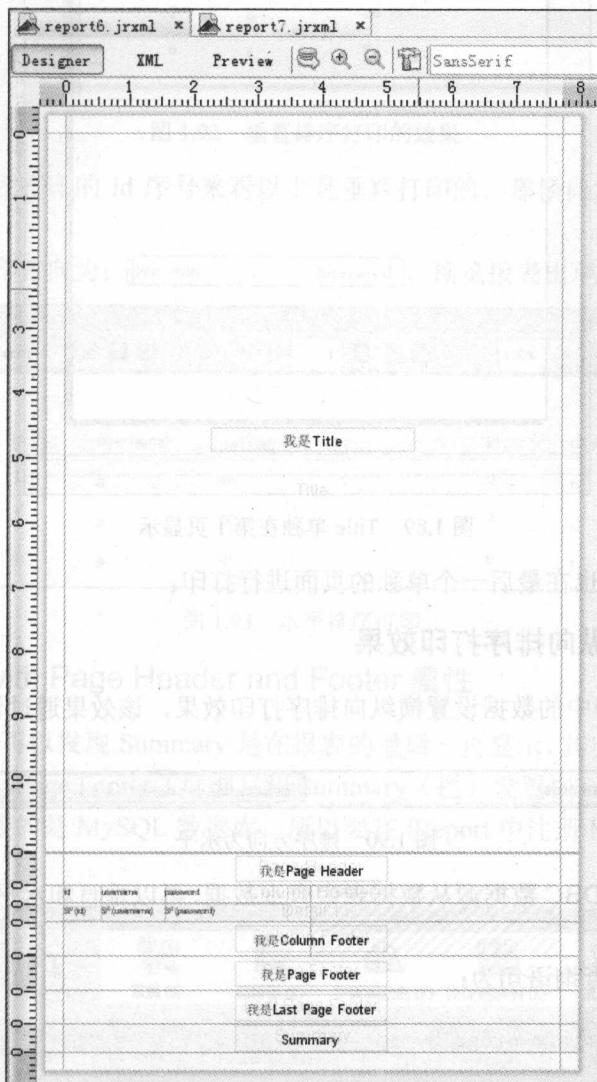


图 1.88 重新布局的 Title 示例

预览效果如图 1.89 所示, 可以看到 Title 单独在第 1 页显示。

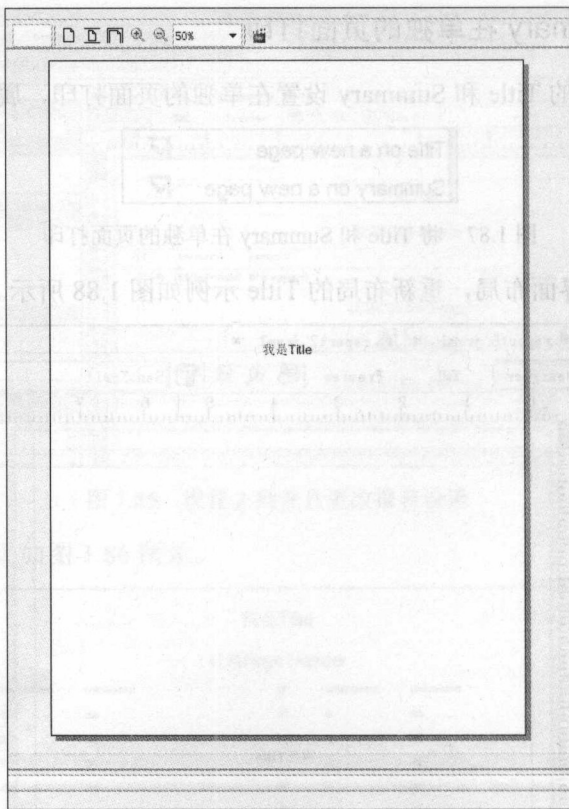


图 1.89 Title 单独在第 1 页显示

而 Summary Band 也在最后一个单独的页面进行打印。

1.8.3 多列横向与纵向排序打印效果

还可以对多列打印中的数据设置横纵向排序打印效果，该效果通过设置如图 1.90 所示的属性来完成。

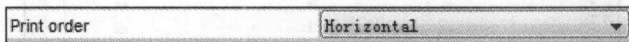


图 1.90 排序方向为水平

本示例需要使用 JDBC 数据源从数据表中查询数据，再以垂直和水平排序的方式显示到报表上。

更改报表的 SQL 查询语句为：

```
select * from userinfo order by id
```

设置打印的排序方向为默认值，也就是垂直方向，效果如图 1.91 所示。

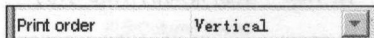
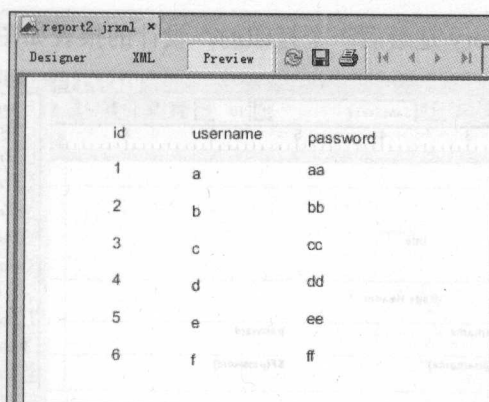


图 1.91 排序方向为垂直

垂直排序的打印效果如图 1.92 所示。

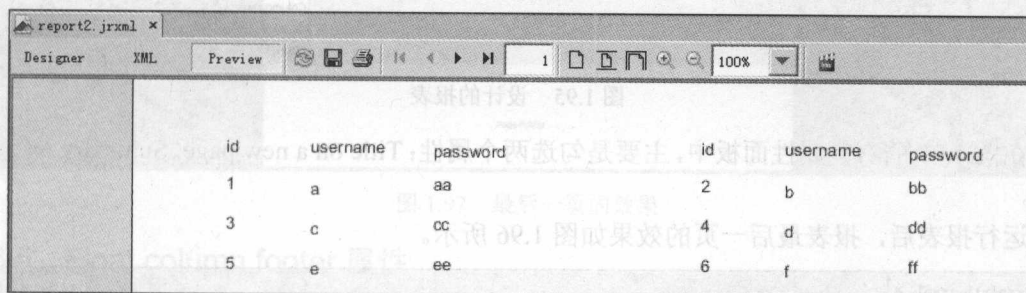


id	username	password
1	a	aa
2	b	bb
3	c	cc
4	d	dd
5	e	ee
6	f	ff

图 1.92 垂直排序打印的效果

从 userinfo 数据表打印的 id 序号来看以上是垂直打印的，那横向水平排序的打印效果如何呢？

继续设置报表打印方向为： ，预览报表出现的效果如图 1.93 所示。



id	username	password	id	username	password
1	a	aa	2	b	bb
3	c	cc	4	d	dd
5	e	ee	6	f	ff

图 1.93 水平排序打印

1.8.4 Summary with Page Header and Footer 属性

在第 1.9.2 小节中可以发现 Summary 是在报表的最后一页显示，其实还可以在最后一页将 Page Header（页头）、Page Footer（页脚）和 Summary（栏）设置在一起显示。

由于本示例要连接的是 MySQL 数据库，所以要在 iReport 中注册 MySQL 的 JDBC 驱动，如图 1.94 所示。

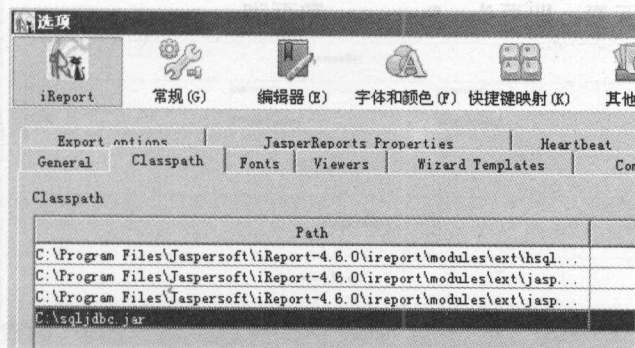


图 1.94 注册 MySQL 驱动

在默认的情况下 Summary 是最后一个打印的，设计的报表如图 1.95 所示。

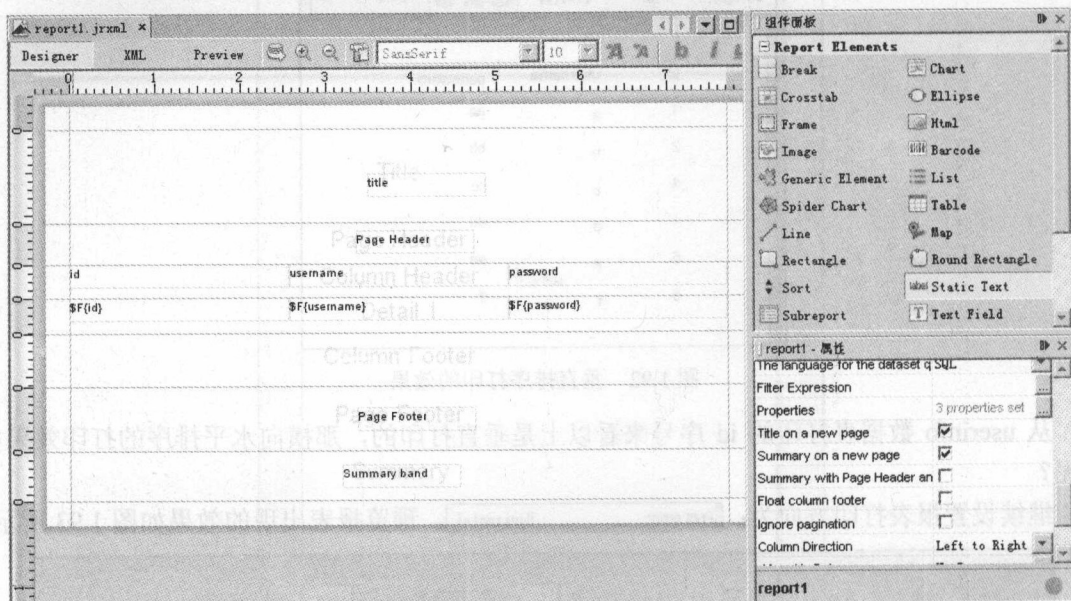


图 1.95 设计的报表

在图 1.95 的右侧属性面板中，主要是勾选两个属性：Title on a new page、Summary on a new page。

运行报表后，报表最后一页的效果如图 1.96 所示。

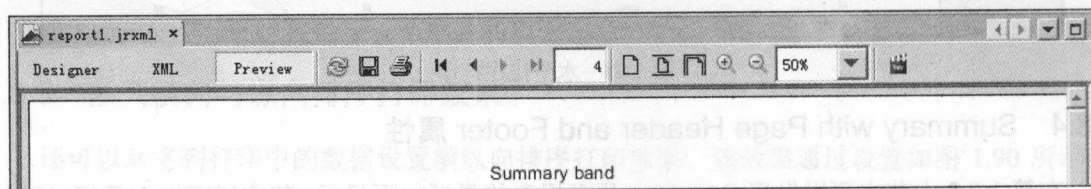


图 1.96 报表最后一页效果

在图 1.96 中仅打印了一个字符串 "Summary band"，页头和页脚并没有打印，如果勾选属性 Summary with Page Header and Footer，将是什么效果呢？效果如图 1.97 所示，可以看出最后一页显示出了 3 个元素，即页头、Summary 和页脚。

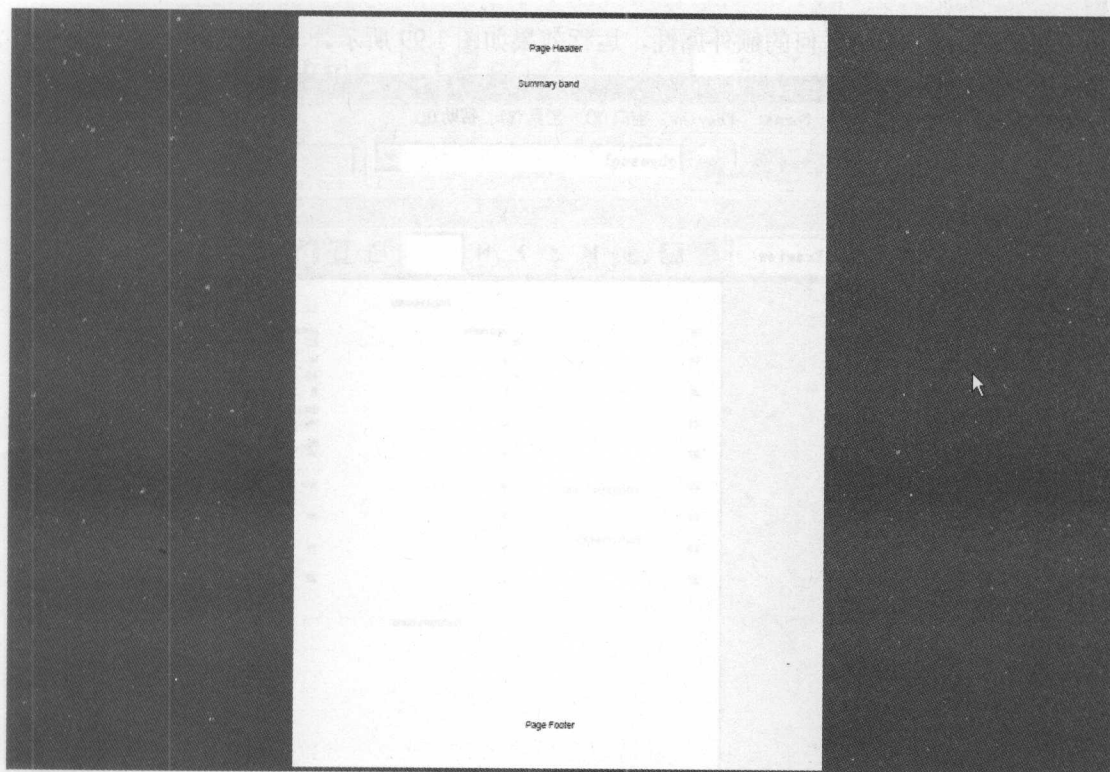


图 1.97 最后一页的效果

1.8.5 Float column footer 属性

属性 Float column footer 的功能是：在最后一页，Column Foot（列脚）是否紧挨着最后一个 Details Band。

设计的报表模板如图 1.98 所示。

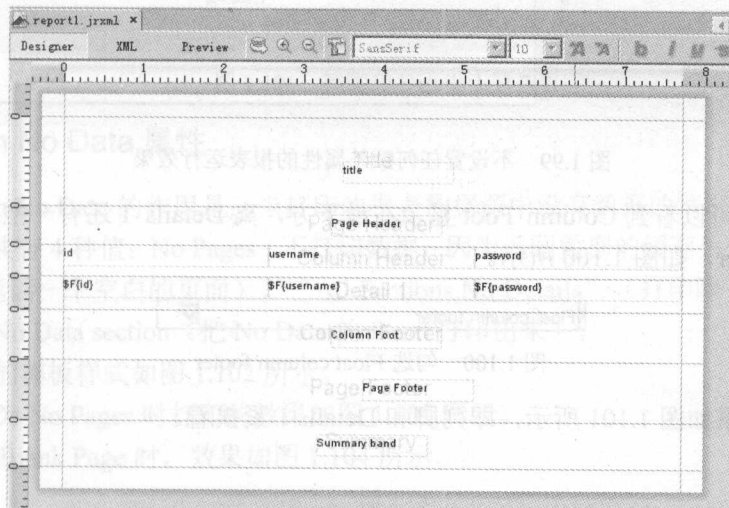


图 1.98 设计的报表模板

在报表中不要添加任何的额外属性，运行效果如图 1.99 所示。

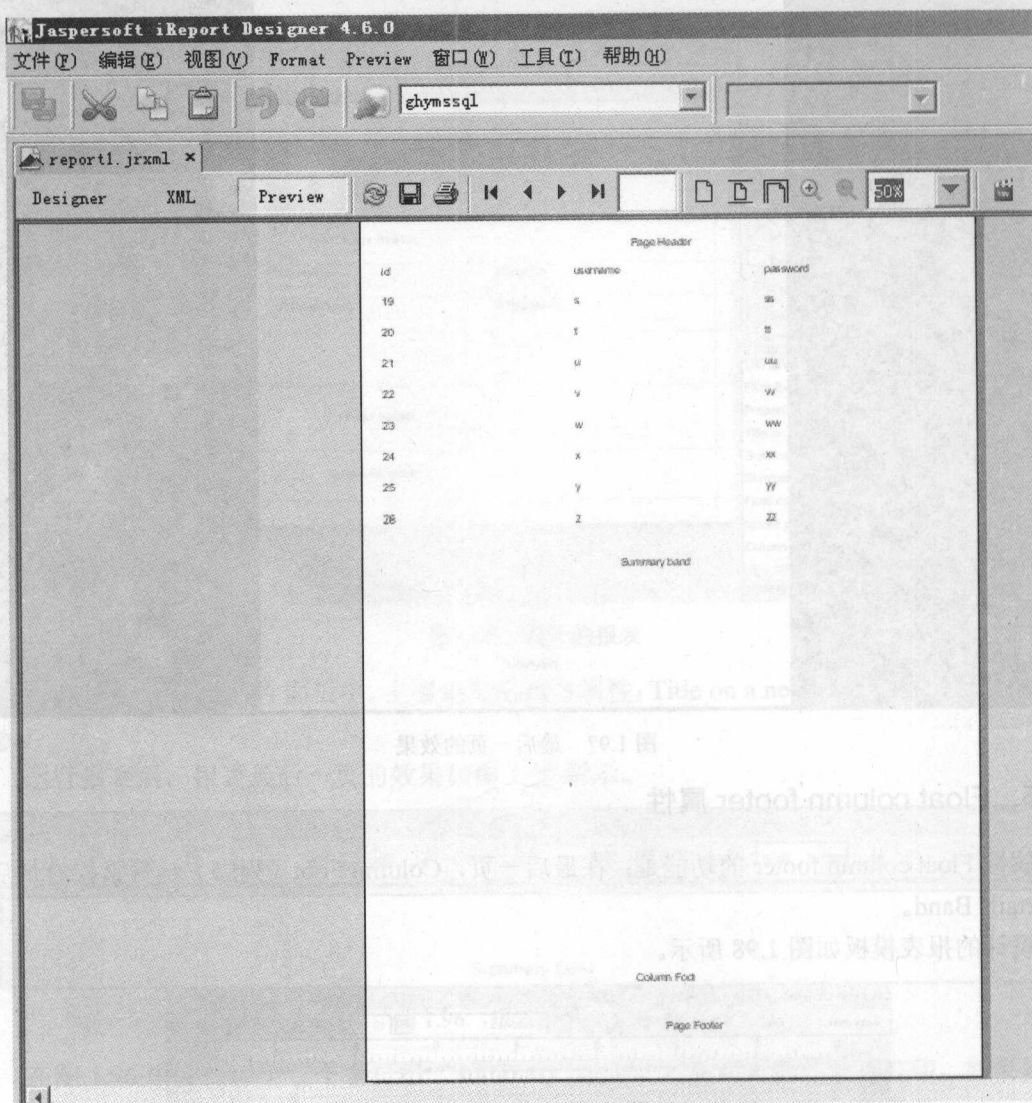


图 1.99 不设置任何额外属性的报表运行效果

从图 1.99 中可以看到 Column Foot 显示在最下方，离 Details 1 还有一些距离，这时勾选 Float column footer，如图 1.100 所示。

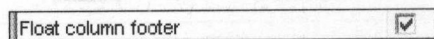


图 1.100 勾选 Float column footer

运行后的效果如图 1.101 所示，即列脚和 Details 1 紧挨着。

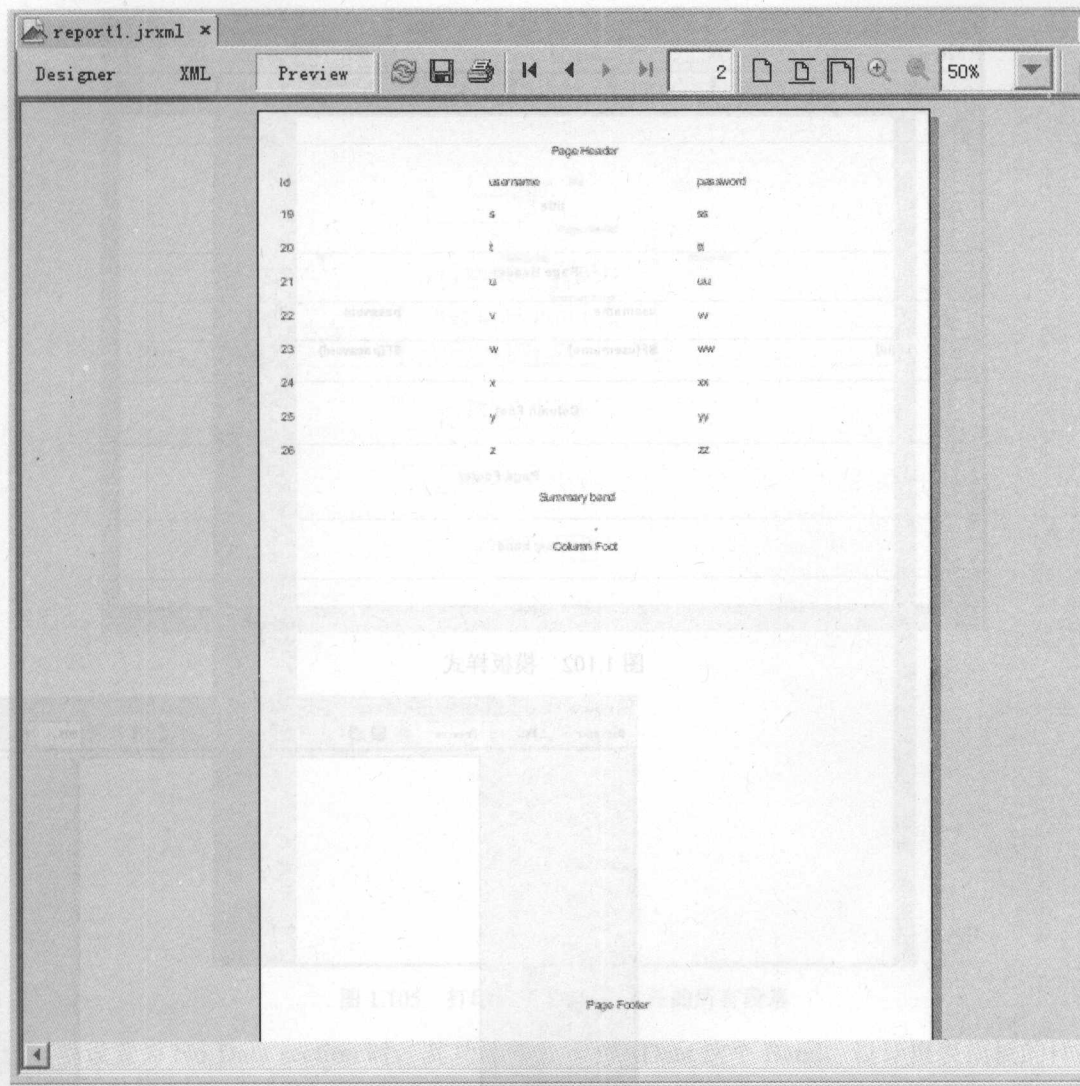


图 1.101 列脚和 Details 1 紧挨着

1.8.6 When No Data 属性

属性 When No Data 的作用是：当打印的报表数据源中没有数据的情况下，报表该如何处理的行为，一共有 4 种值：No Pages（不打印数据，因为返回数据的缓存为空，这是默认值），Blank Page（返回一个空白的页面），“All Sections, No Details”（打印除了 Details 之外的所有的段落），No Data section（把 No Data 的 Band 打印出来）。

设计报表的模板样式如图 1.102 所示。

当值设置为 No Pages 时打印的效果如图 1.103 所示。

当设置为 Blank Page 时，效果如图 1.104 所示。

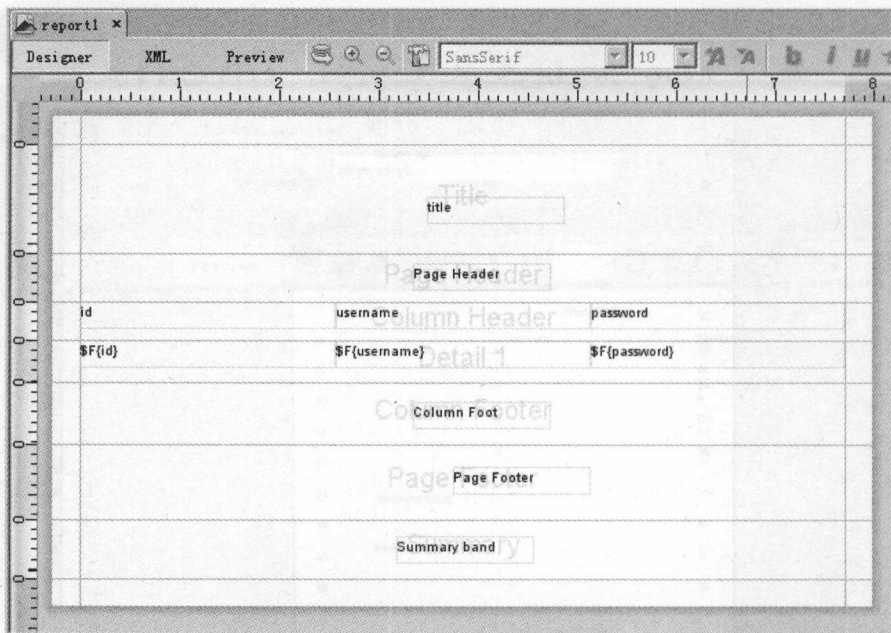


图 1.102 模板样式

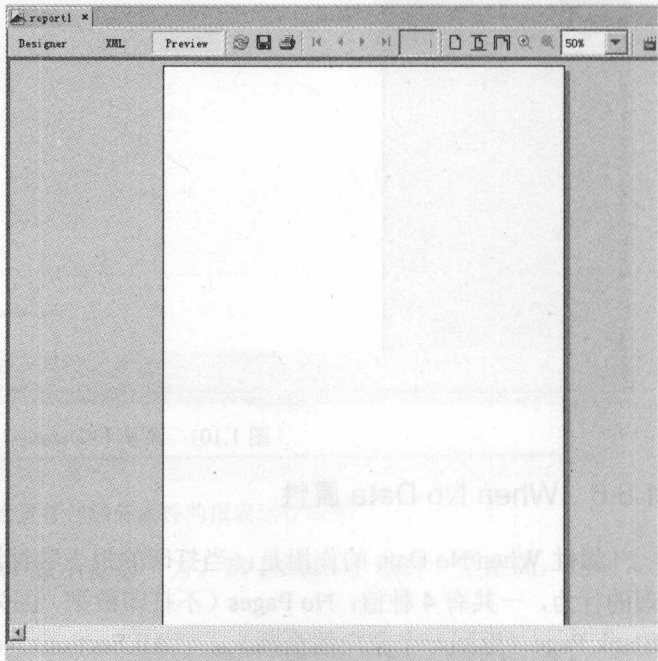


图 1.104 空白页

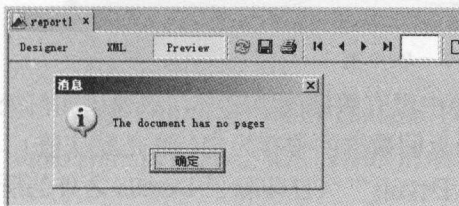


图 1.103 无数据

当设置为 “All Sections, No Details” 时，效果如图 1.105 所示。

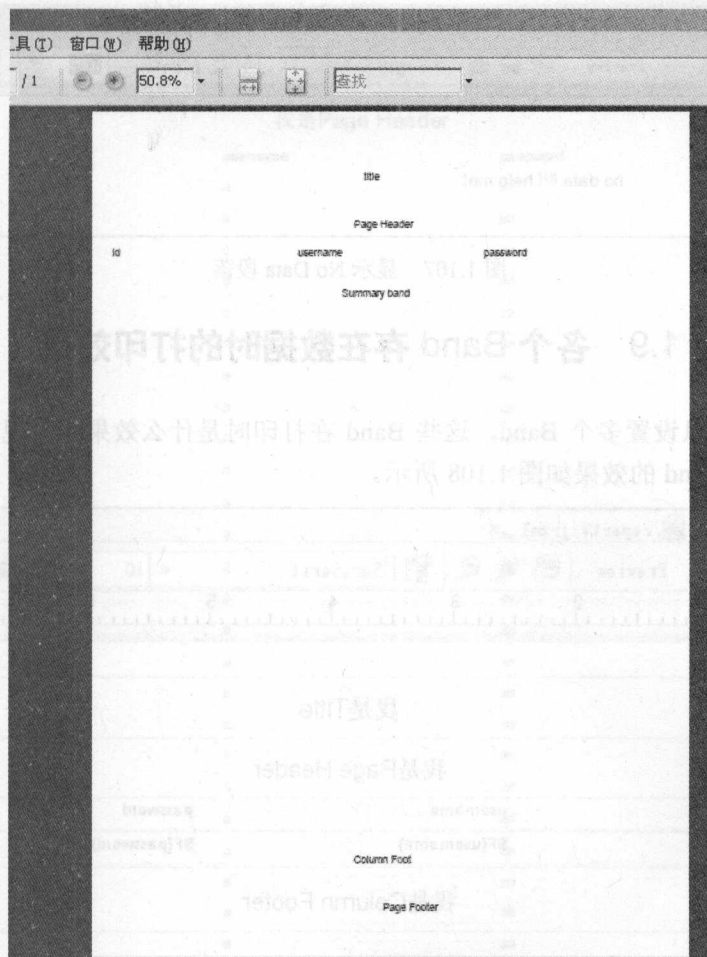


图 1.105 打印除了 Details 之外的所有段落

当设置为 No Data section 时，其功能是显示 No Data 这个 Band，设计报表模板的样式如图 1.106 所示。

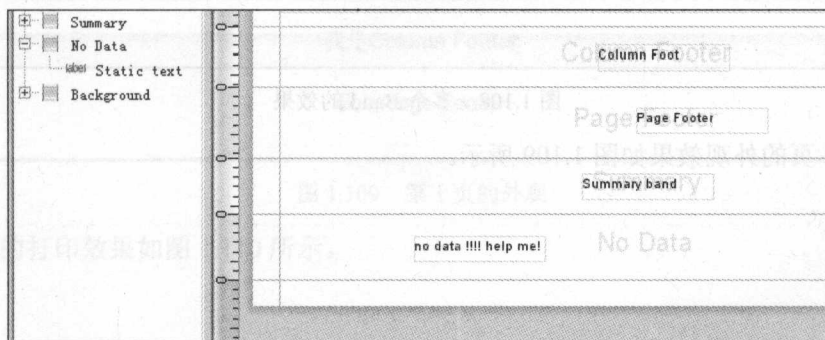


图 1.106 添加 No Data 段落

显示 No Data 段落的效果如图 1.107 所示。

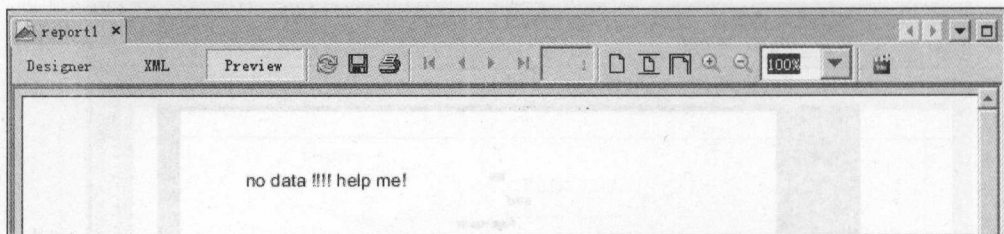


图 1.107 显示 No Data 段落

1.9 各个 Band 存在数据时的打印效果

在.jrxml 中可以设置多个 Band，这些 Band 在打印时是什么效果呢？现设计如下的.jrxml 外观样式，多个 Band 的效果如图 1.108 所示。

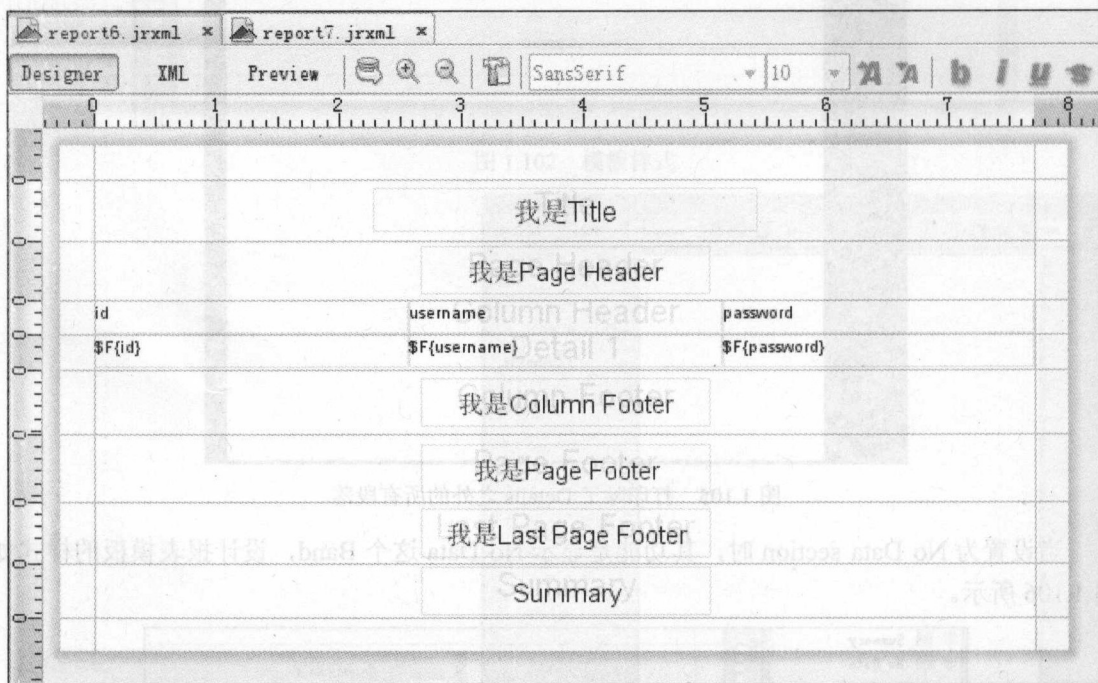


图 1.108 多个 Band 的效果

报表第 1 页的外观效果如图 1.109 所示。

我是Title		
我是Page Header		
id	username	password
1	a	aa
2	b	bb
3	c	cc
4	d	dd
5	e	ee
6	a	aa
7	a	aa
8	a	aa
9	a	aa
10	a	aa
11	a	aa
12	a	aa
13	a	aa
14	a	aa
15	a	aa
16	a	aa
17	a	aa
18	a	aa
19	a	aa
20	a	aa
21	a	aa
22	a	aa
23	a	aa
24	a	aa
25	a	aa
26	a	aa
27	a	aa
28	a	aa
29	a	aa
30	a	aa
我是Column Footer		
我是Page Footer		

图 1.109 第 1 页的外观

第 2 页的打印效果如图 1.110 所示。

我是Page Header		
id	username	password
31	a	aa
32	a	aa
33	a	aa
34	a	aa
35	a	aa
36	a	aa
37	a	aa
38	a	aa
39	a	aa
40	a	aa
41	a	aa
42	a	aa
43	a	aa
44	a	aa
45	a	aa
46	a	aa
47	a	aa
48	a	aa
49	a	aa
50	a	aa
51	a	aa
52	a	aa
53	a	aa
54	a	aa
55	a	aa
56	a	aa
57	a	aa
58	a	aa
59	a	aa
60	a	aa
61	a	aa
我是Column Footer		
我是Page Footer		

图 1.110 第 2 页的效果

最后一页的效果如图 1.111 所示。

我是Page Header		
id	username	password
62	a	33
63	a	33
64	a	33
65	a	33
66	a	33
67	a	33
68	a	33
69	a	33
70	a	33
71	a	33
72	a	33
73	a	33
74	a	33
75	a	33
76	a	33
77	a	33
78	a	33
79	a	33
80	a	33
81	a	33
82	a	33
83	a	33
84	a	33
Summary		
我是Column Footer		
我是Last Page Footer		

图 1.111 最后一页的打印效果

第 2 章 控 件

↓ 导言

本章利用大量的篇幅讲述报表控件及控件的常用属性，使用这些属性是熟练开发报表的基础，读者应该着重掌握如下知识点：

- ★ 控件的对齐
- ★ 控件的位置
- ★ Image 控件的使用
- ★ 文本控件的使用
- ★ Frame 框架控件的使用
- ★ PositionType 属性的使用
- ★ StretchType 属性的使用

2.1 控件的常用知识

iReport 中的控件列表如图 2.1 所示。

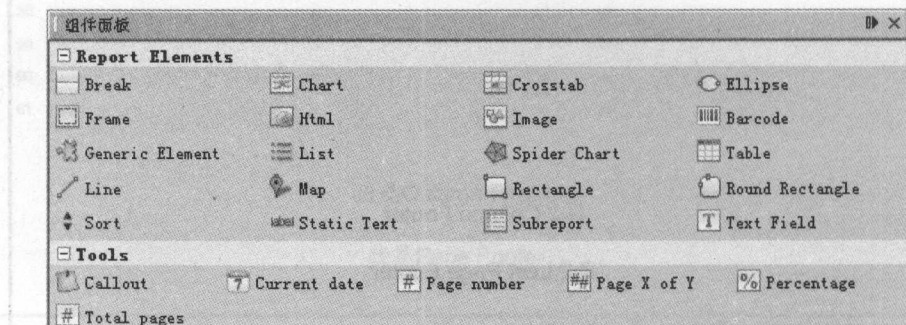


图 2.1 iReport 中的控件

iReport 中的控件主要包括：Line, Rectangle, Ellipse, Static Text, Text Field, Image, Frame, Subreport, Crosstab, Chart, Break。

其中，Static Text 控件用于显示静态的文本字符，不可以添加 Java 的运算表达式。

Break 控件用于强制分页，可以在 Detail 1 中设置如下结构的报表，效果如图 2.2 所示。

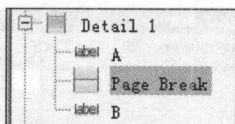


图 2.2 添加强制分页 Break 的报表结构

报表模板外观如图 2.3 所示。

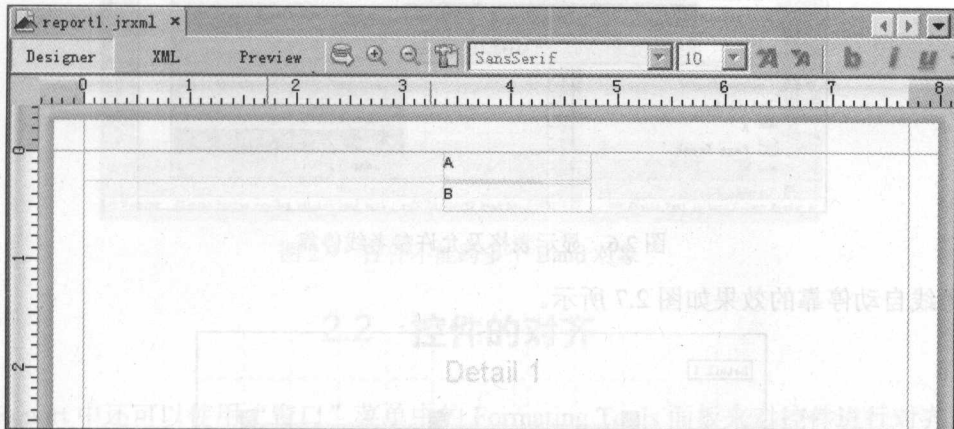


图 2.3 强制分页的模板文件外观

报表模板预览后的效果如图 2.4 所示。

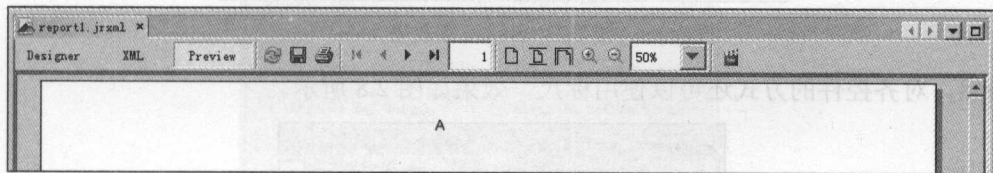


图 2.4 A 在第 1 页

字符 B 在第 2 页，效果如图 2.5 所示。

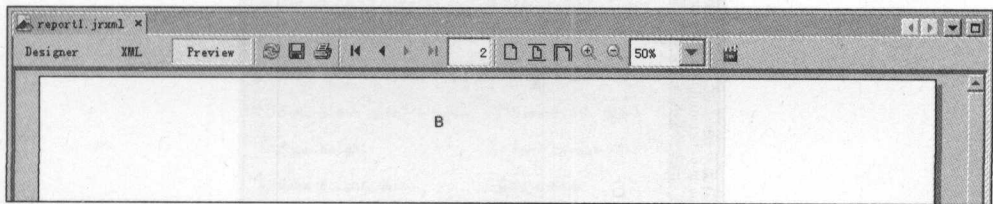


图 2.5 B 在第 2 页

将控件放入报表设计器时，在选中控件后可以按上下左右方向键来以 1px 级别进行控件的移动，还可以按下 Shift 键结合上下左右键进行每隔 10px 的移动。

为了对控件的位置进行整体风格对齐，建议勾选 Show Grid 和 Snap To Grid 菜单，这样可以实现报表设计器的表格显示，表格的边框其实就是对齐的参考线，并且可以将控件自动停靠到表格的参考线上，效果如图 2.6 所示。



图 2.6 显示表格及允许参考线停靠

参考线自动停靠的效果如图 2.7 所示。

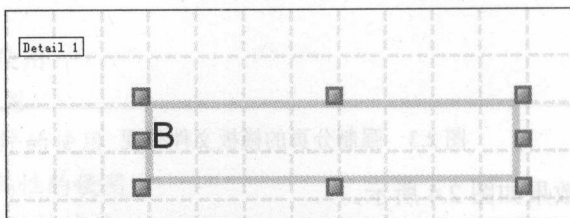


图 2.7 自动停靠效果

当然，对齐控件的方式还可以使用标尺，效果如图 2.8 所示。

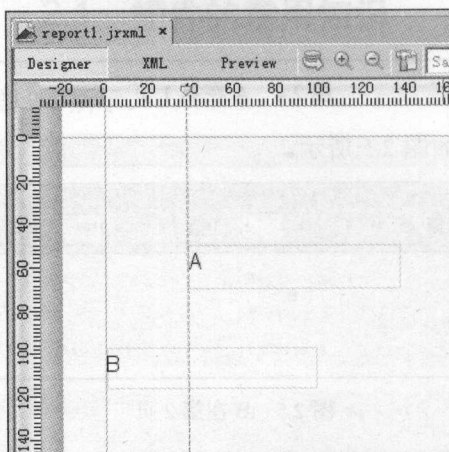


图 2.8 使用标尺

需要注意的是，标尺的功能和 Snap To Grid 是互斥的，同一时间只能使用一种。如果这两种功能都允许使用，则默认使用 Snap To Grid 功能。

另外，控件不能跨多个 Band 对象，如果这样操作，则在预览时会出现提示，效果如图 2.9 所示。

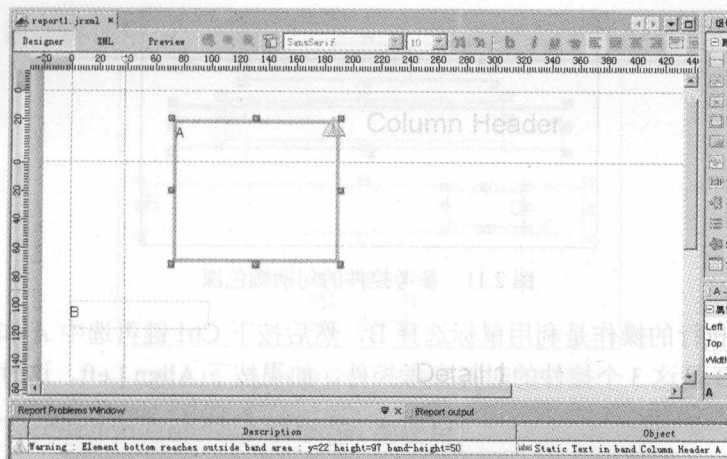


图 2.9 控件不能跨多个 Band 对象

2.2 控件的对齐

在 iReport 中还可以使用“窗口”菜单中的 Formating Tools 面板来对控件进行对齐。对齐的选项如图 2.10 所示。

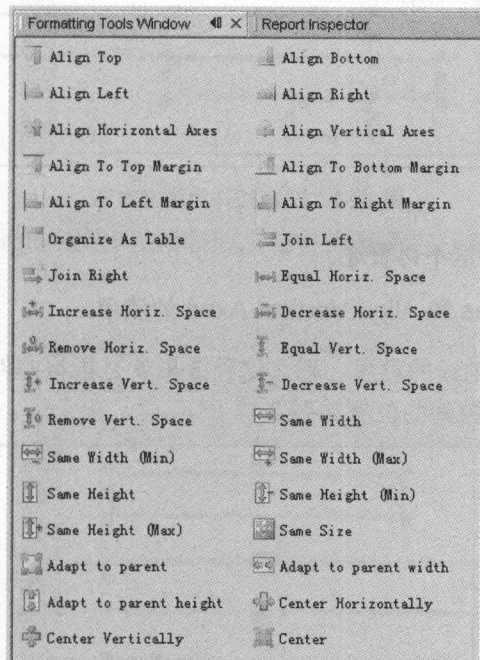


图 2.10 布局对齐的种类

1. Align Top/Align Bottom/Align Left/Align Right 的使用

此种对齐方式是指：如果有多个控件被选中时这些控件的对齐方式。但在这里需要注意的是，有一个参考控件的概念，第 1 个被选中的控件就是参考控件，如图 2.11 所示。

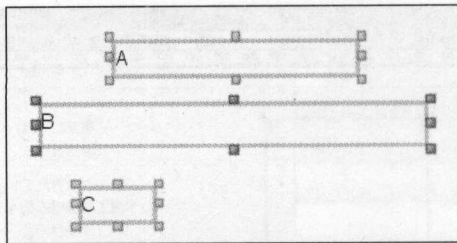


图 2.11 参考控件的句柄颜色深

在图 2.11 中执行的操作是利用鼠标选择 B，然后按下 Ctrl 键再选中 A 和 C，这时 B 第 1 个被选中，所以 B 是这 3 个控件的对齐参考控件，如果按下 Align Left，这时的效果如图 2.12 所示。

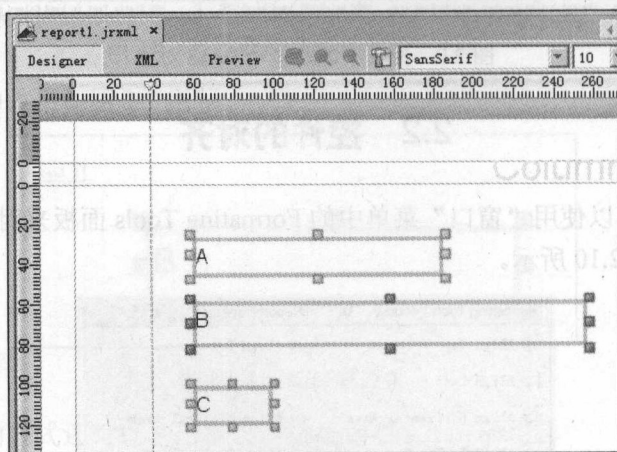


图 2.12 以 B 控件为准左对齐

其他的 3 个对齐方式这里不再赘述。

2. Align Horizontal Axes 和 Align Vertical Axes 的使用

单词 Axes 是轴心、中轴线的意思，那么这种效果就是指将控件的高度除以 2 即为中轴，也可以将宽度除以 2，布局控件的位置如图 2.13 所示。

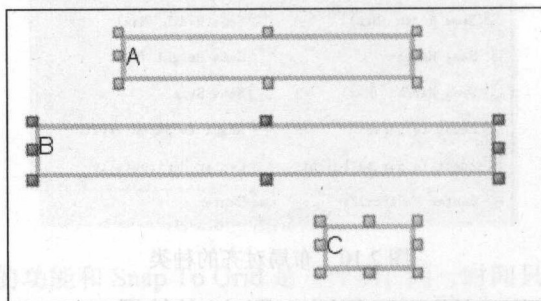


图 2.13 B 控件是参考控件

单击 Align Vertical Axes 控件，意思是将这 3 个控件的垂直对齐方式设置为中轴线对齐，

单击后的效果如图 2.14 所示。

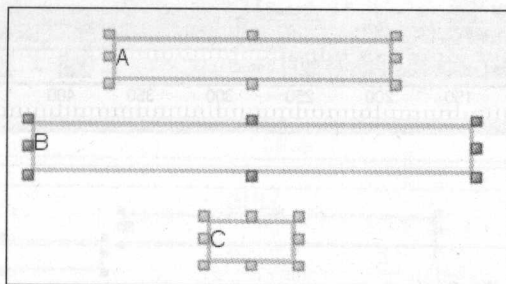


图 2.14 以 B 控件的中轴线为准进行对齐

3. Align To Top/Bottom/Left/Right Margin 的使用

Align To Top/Bottom/Left/Right Margin 的功能是：定义控件以报表边距为对齐方式。设置的报表模板内容如图 2.15 所示。

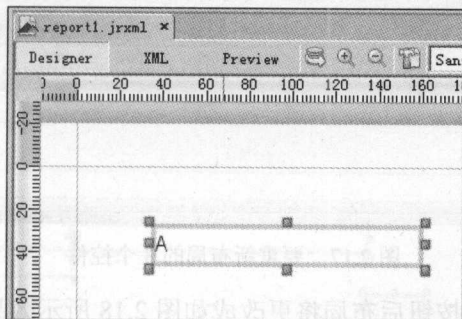


图 2.15 只有 A 的模板

单击 Align To Left Margin 按钮，以报表的左边界为参考进行对齐，单击后的效果如图 2.16 所示。

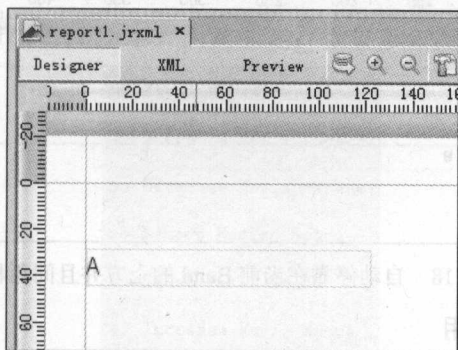


图 2.16 以报表左边距为参考

4. Organize As Table 的使用

Organize As Table 按钮的功能是：可以使多个被选中的控件在当前的 Band 上方进行 Top 对齐，它经常用在将当前要打印的数据列进行自动停靠并且间距相同的情况下。布局报表模板

的界面效果如图 2.17 所示。

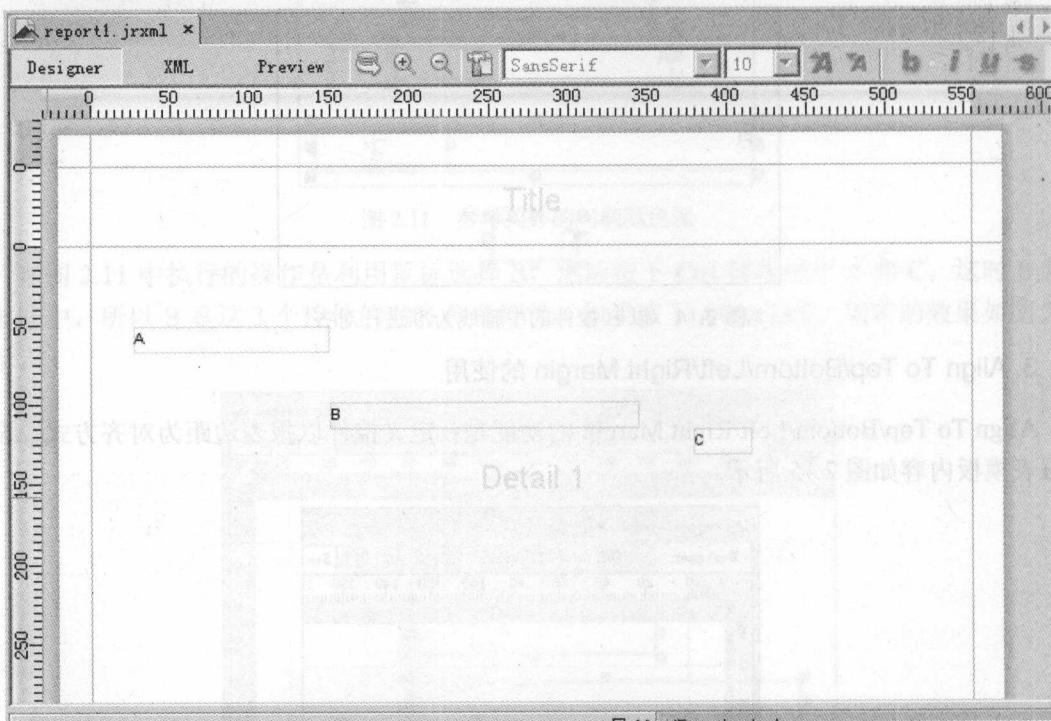


图 2.17 要重新布局的 3 个控件

单击 **Organize As Table** 按钮后布局将更改成如图 2.18 所示效果，即自动停靠在当前 Band 的上方，并且间距相同。

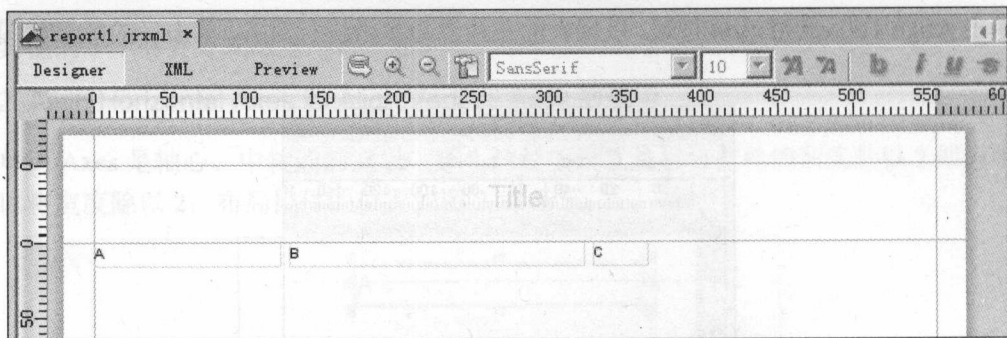


图 2.18 自动停靠在当前 Band 的上方并且间距相同

5. Join Left/Right 的使用

Join Left/Right 按钮的功能是：设置多个控件首尾垂直相接的效果，报表模板的设计如图 2.19 所示。

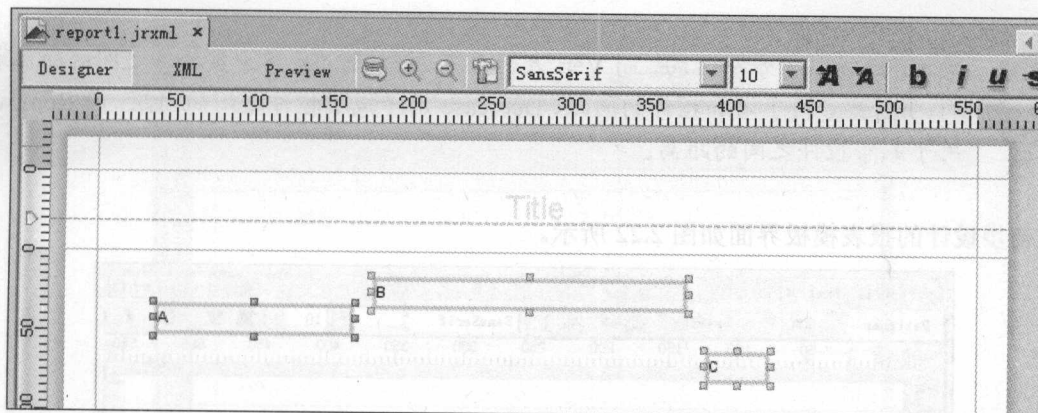


图 2.19 设计报表模板

单击 Join Left 按钮，A、B 和 C 就首尾垂直相接了，效果如图 2.20 所示。

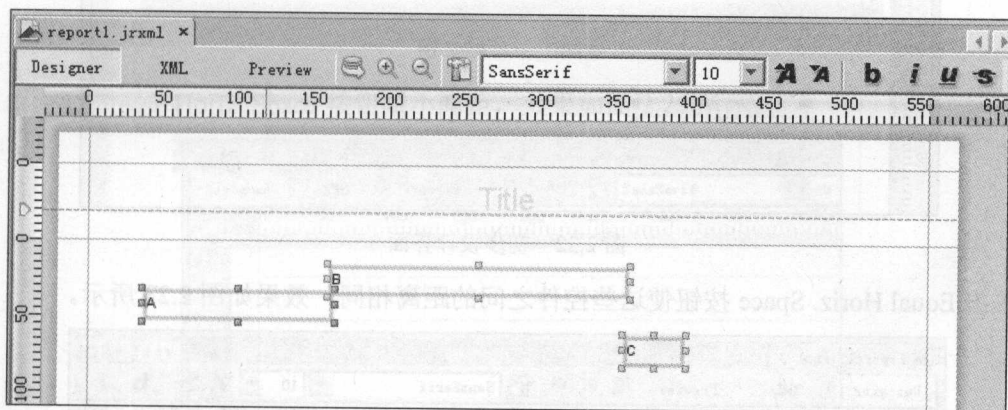


图 2.20 首尾垂直相接的效果

6. Equal/Increase/Decrease/Remove Horiz. Space 的使用

8 个间距设置按钮（如图 2.21 所示）的功能是：设置多个控件之间的间距。

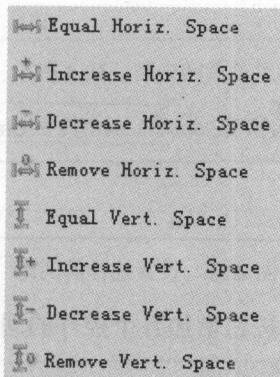


图 2.21 8 个间距设置按钮



提示

单词 Horiz 表示水平方向，而 Vert 表示垂直方向。Equal 用于使控件之间的距离相同，Increase 用于增加控件的间距，Decrease 用于减小控件之间的间距，而 Remove 用于删除控件之间的距离。

初步设计的报表模板界面如图 2.22 所示。

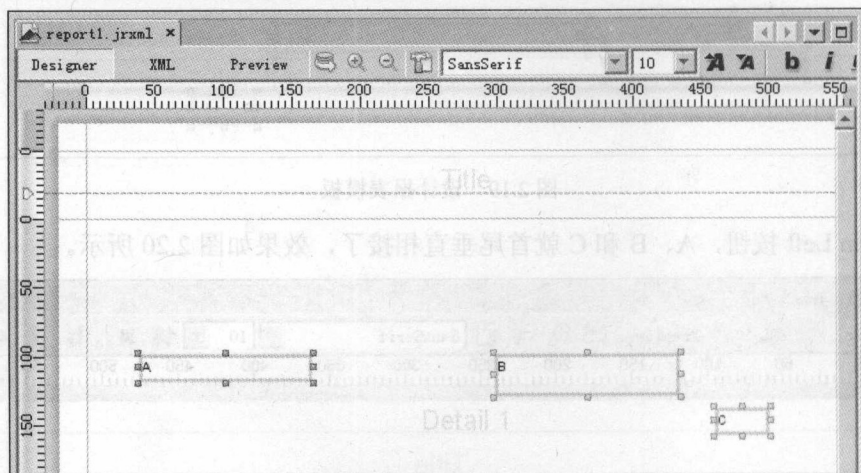


图 2.22 初步设计界面

单击 Equal Horiz. Space 按钮使这些控件之间的距离相同，效果如图 2.23 所示。

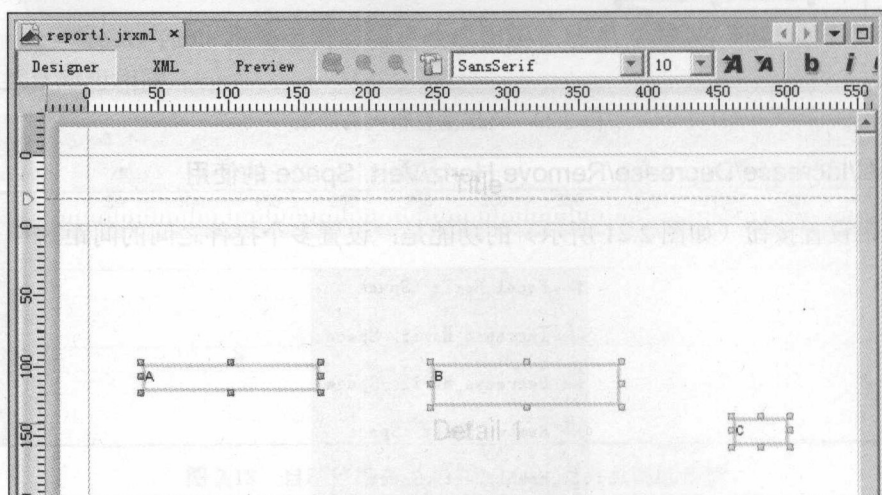


图 2.23 设置控件的间距

继续单击 Increase Horiz. Space 按钮增加各个控件的间距，它们的间距永远保持相同，效果如图 2.24 所示。

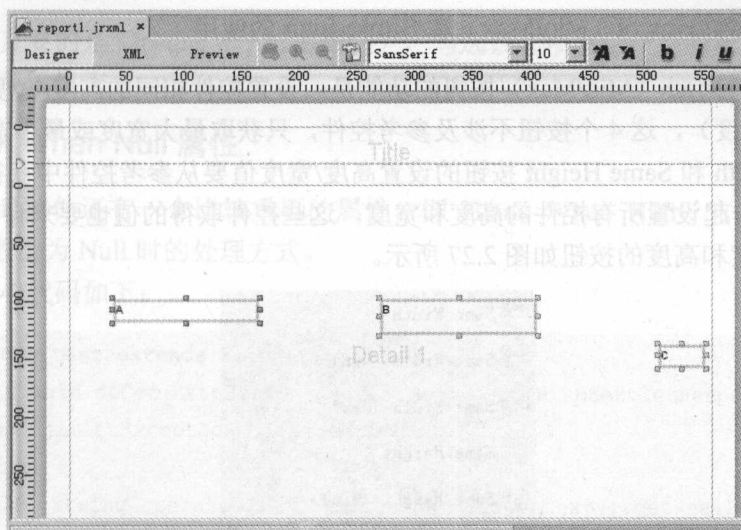


图 2.24 增加间距

单击 Decrease Horiz. Space 按钮减小间距，效果如图 2.25 所示。

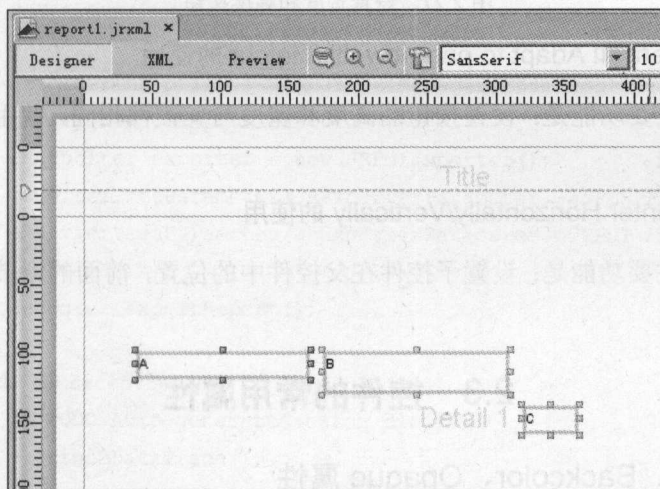


图 2.25 减少间距效果

还可以删除间距，即单击 Remove Horiz. Space 按钮，效果如图 2.26 所示。

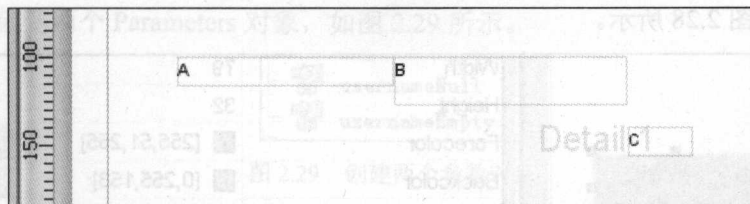


图 2.26 删除间距

上面的示例是在水平方向上进行演示，垂直的效果相似，这里不再赘述。

7. Same Width/Height (Min/Max) 和 Same Size 的使用

Same Width/Height (Min/Max) 按钮的功能是：设置控件相同的高度/宽度（取决于最大/小控件的高度/宽度），这 4 个按钮不涉及参考控件，只获取最大宽度或最小宽度的值。

而 Same Width 和 Same Height 按钮的设置高度/宽度值要从参考控件中获得。当然还可以使用 Same Size 一起设置所有控件的高度和宽度，这些控件取得的值也要来自于参考控件。

7 个设置宽度和高度的按钮如图 2.27 所示。

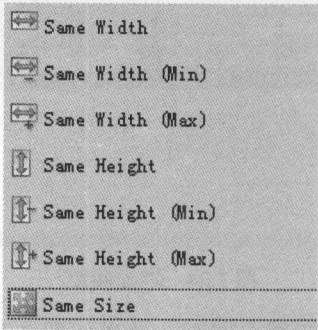


图 2.27 设置宽度和高度按钮

8. Adapt to parent 和 Adapt to parent width/Height 的使用

这 3 个按钮的主要功能是：设置按钮的高度和宽度与父控件相同，使用起来非常简单，有些类似于填充的效果。

9. Center 和 Center Horizontally/Vertically 的使用

这 3 个按钮的主要功能是：设置子控件在父控件中的位置，前面的章节已使用过，这里不再赘述。

2.3 控件的常用属性

2.3.1 Forecolor、Backcolor、Opaque 属性

在这里可以设置控件的外观颜色，主要使用 3 个属性：Forecolor 用于设置文字颜色；Backcolor 用于设置背景颜色；Opaque 用于设置是否透明。例如，设置控件文字颜色为粉色、背景为绿色，如图 2.28 所示。

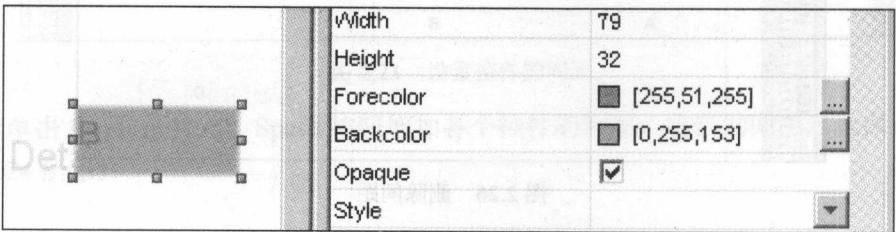


图 2.28 控件文字颜色为粉色、背景为绿色

其中比较重要的是 Opaque（不透明属性），勾选该选项表示控件的背景不透明，这时即可将背景颜色显示出来了。

2.3.2 Blank When Null 属性

Text Field 控件还有一个比较重要的属性，即 Blank When Null，它的功能是：设置当 Parameters 参数值为 Null 时的处理方式。

Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            String operateDir = this.getServletContext().getRealPath("/");
            String jasperFilePath = operateDir + "report4.jasper";
            String pdfFilePath = operateDir + "report4.pdf";
            HashMap param = new HashMap();
            param.put("usernameNull", null);
            param.put("usernameEmpty", "");
            JasperPrint print = JasperFillManager.fillReport(jasperFilePath,
                param, new JREmptyDataSource());
            JRExporter exporter = new JRPdfExporter();
            exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,
                pdfFilePath);
            exporter.exportReport();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

在报表中创建两个 Parameters 对象，如图 2.29 所示。

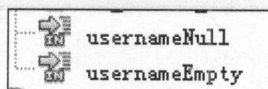


图 2.29 创建两个参数对象

设置这两个参数对象的 Blank When Null 属性为不勾选状态，如图 2.30 所示。

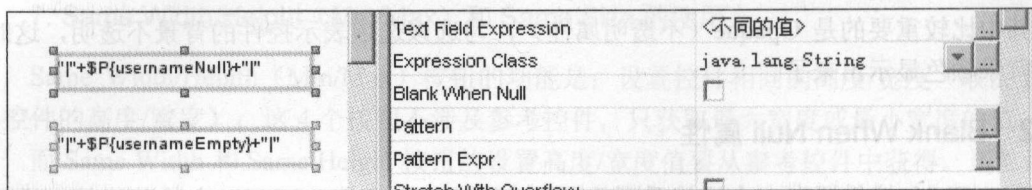


图 2.30 设置为不勾选

程序运行后导出 PDF 文件，内容如图 2.31 所示。

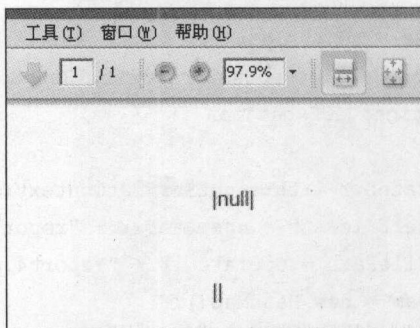


图 2.31 显示出了 "null" 字符串

可是有些时候不希望显示为 "null" 字符串，继续设置属性，如图 2.32 所示。

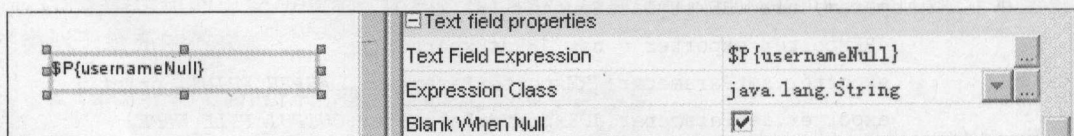


图 2.32 设置 Text Field Expression 值为 \$P{usernameNull}

再继续设置属性，如图 2.33 所示。

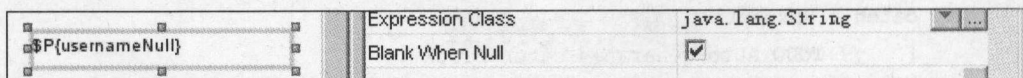


图 2.33 设置为勾选状态

再次运行程序，如图 2.34 所示。

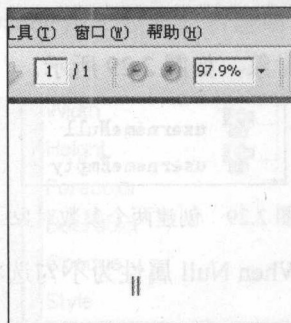


图 2.34 "null" 字符串没有了

从图 2.34 中可以看到空字符串并没有打印出来。

2.3.3 Position Type 属性

属性 Position Type 的功能很重要,因为连接数据源后在 Band 中显示数据时,数据量的多少不固定,也就是 Band 的高度一直在变化,但又想在报表中显示一些附加的元素,这时这个元素的位置就可以使用此属性进行定义。

向 MySQL 数据表中添加数据,共两条记录,其中第 2 条记录中的 password 字段内容文本较多,字段类型为 longtext。

设计报表模板,内容如图 2.35 所示。

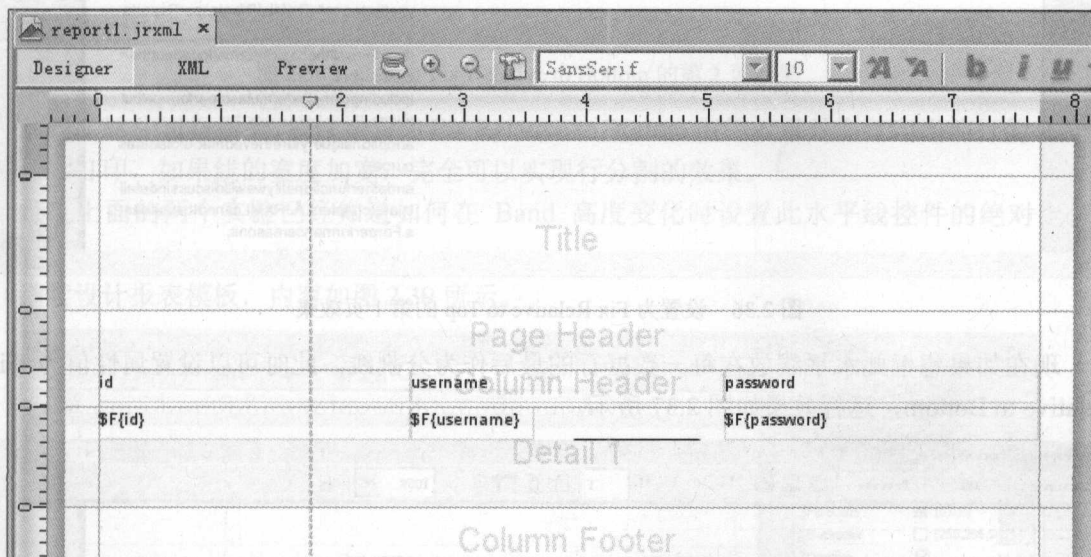


图 2.35 设计报表模板

由于第 2 条记录的 password 字段较多,所以勾选 password 控件的属性 ☒ Stretch With Overflow,即 Text Field 控件的高度随着文本的多少而自动伸展。

从图中可以看到,有一条水平线,这条水平线与 username 和 password 控件在垂直和水平方向上并没有交错,现在想实现的效果是:这条水平线一直保持与 Detail 1 的 Top 距离,设置 Position Type 的值为 Fix Relative to Top,它也是此属性的默认值,即在报表运行时,向 username 和 password 填充数据时,Band 的高度随着文本的多少在不断变化,而此水平线一直保持在设计报表时与 Detail 1 的 Top 距离,运行效果如图 2.36 所示。

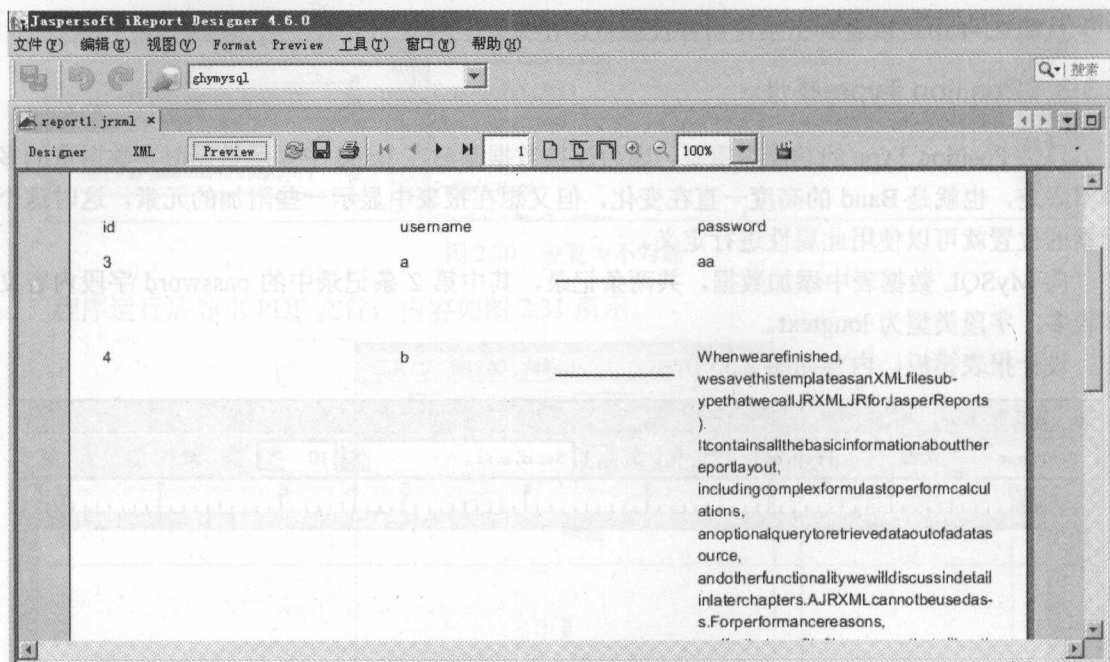


图 2.36 设置为 Fix Relative to Top 的第 1 页效果

现在如果想把此水平线放在每一数据行的最后作为分割线，此时可以设置属性值为 Fix Relative to Bottom，运行效果如图 2.37 所示。

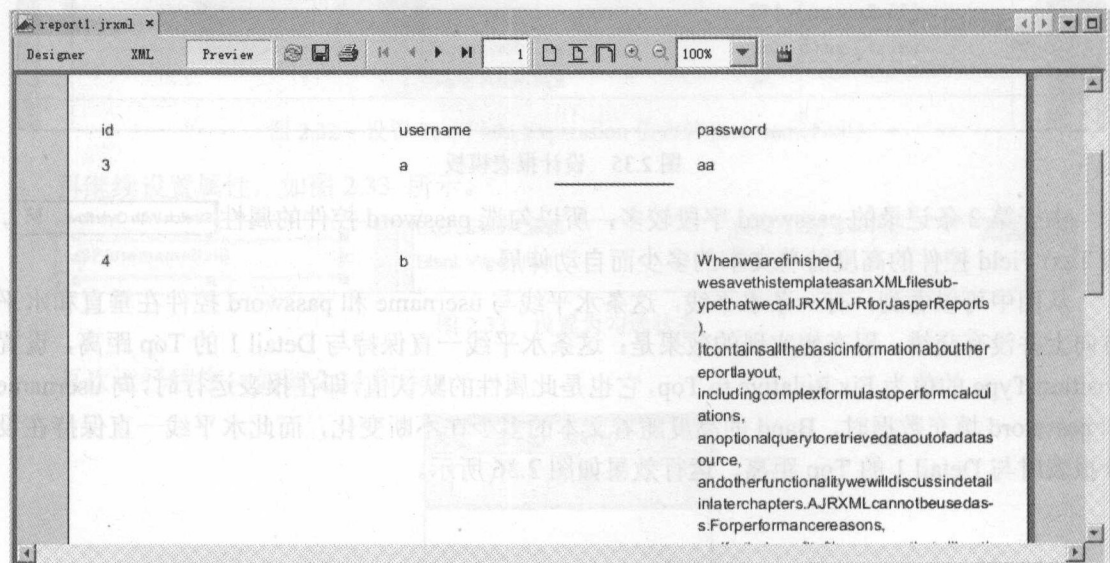


图 2.37 设置为 Fix Relative to Bottom 的第 1 页效果

再来查看如图 2.38 所示的效果。

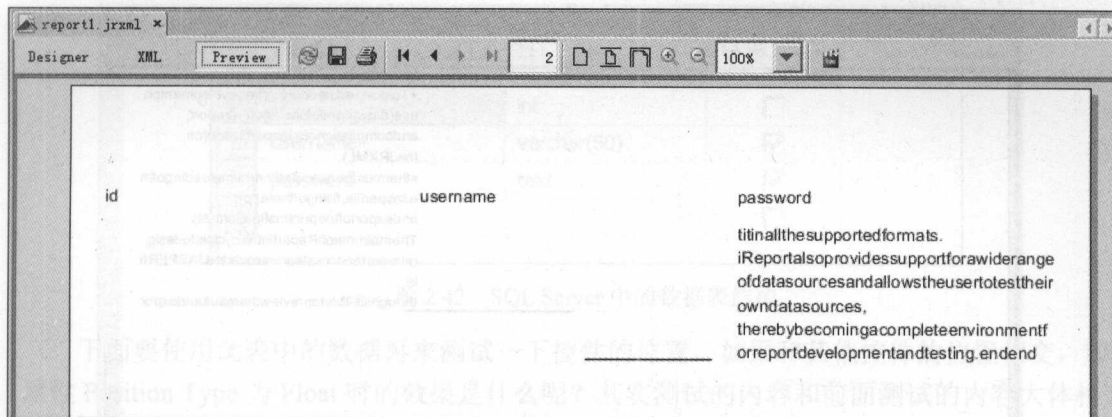


图 2.38 设置为 Fix Relative to Bottom 的第 2 页效果

从图 2.38 中可以看到,属性设置为 Fix Relative to Bottom 值的效果是在打印当前行的 Band 最后进行打印,如果线的宽度加宽,完全可以实现行分割的效果。

经过上面的两个实验已经知道如何在 Band 高度变化时设置此水平线控件的绝对定位问题。

继续设计报表模板,内容如图 2.39 所示。

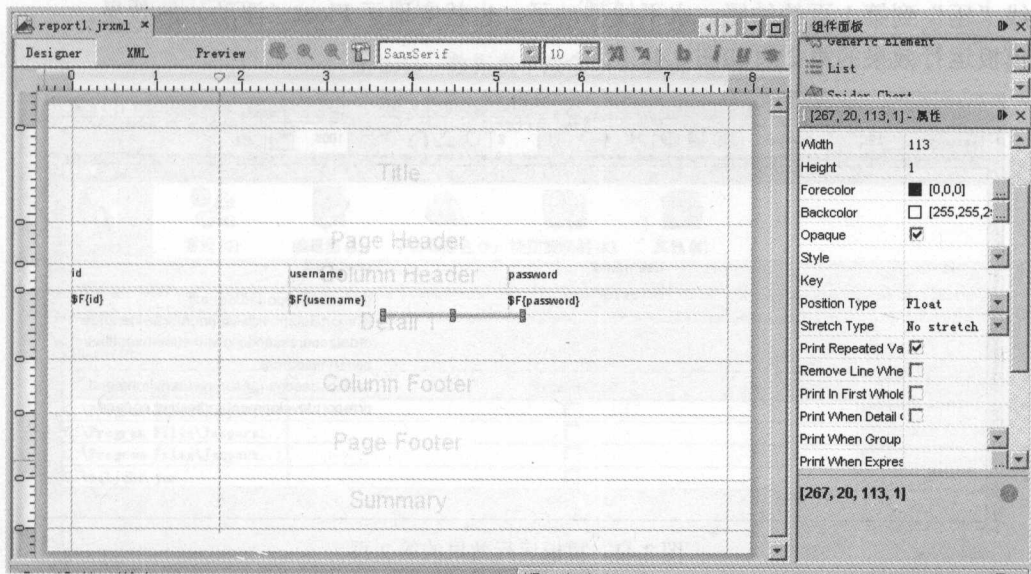


图 2.39 模板改动效果

在如图 2.39 所示的效果中可以看到 `$F{password}` 和水平线的垂直距离没超过 5px, 这时水平线是打印在当前页的底部。

将水平线控件的 Position Type 属性值设置为 Float (浮动), 模板运行效果的第 1 页如图 2.40 所示。

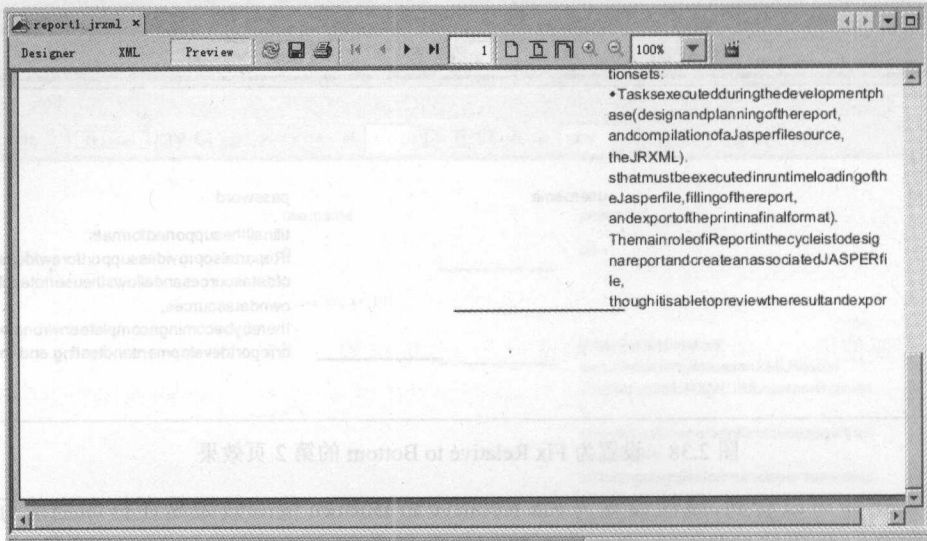


图 2.40 模板运行效果的第 1 页

从图 2.40 中可以看到，水平线真的是“Float（浮动）”了，被放在了第 1 页的最后，由于 password 字段显示出来的数据过多，而又设置水平线的属性值为 Float，所以 password 值把水平线“压”到第 1 页的结尾，水平线移动了，也就实现了 Float（浮动）的效果。

模板运行效果的第 2 页如图 2.41 所示。

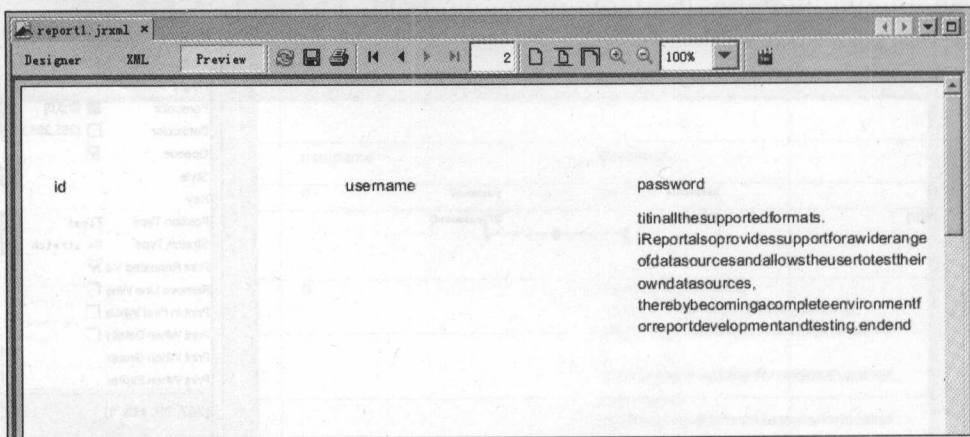


图 2.41 模板运行效果的第 2 页

从以上两幅图中可以看到，设置为 Float 属性后，当 Band 的高度在一页中放不下时，水平线被放置在 Band 打印的第 1 页的最后位置进行显示（前提是 password 控件和水平线的垂直距离要超过 5px），所以 Float 使用的情况比较少，而 Fix Relative to Bottom 属性值经常用在此 Band 打印完毕后添加一个水平分割线时使用。

上面的示例是在报表中连接 MySQL 数据源，其实 iReport 还可以连接 SQL Server 数据源，在 SQL Server 中创建的数据表结构如图 2.42 所示。

TC07\SQL200... dbo. userinfo		TC07\SQL200... dbo. userinfo	
	列名	数据类型	允许 Null 值
▶	id	int	<input type="checkbox"/>
	username	varchar(50)	<input checked="" type="checkbox"/>
	password	text	<input checked="" type="checkbox"/>
			<input type="checkbox"/>

图 2.42 SQL Server 中的数据表结构

下面要使用此表中的数据再来测试一下控件的位置，如果和其他控件的位置相交，那么，属性 Position Type 为 Float 时的效果是什么呢？其实测试的内容和前面测试的内容大体相同，只是使用的数据源不同。

数据表内容如图 2.43 所示，可以看到数据表字段值以 endend 结尾。

TC07\SQL200... dbo. userinfo		TC07\SQL200... dbo. userinfo	
	id	username	password
	1	a	aa
▶	2	b	最后更新和压缩文件名的信息。endend
*	NULL	NULL	NULL

图 2.43 数据表字段值以 endend 结尾

在 iReport 中添加 SQL Server JDBC 驱动，如图 2.44 所示。

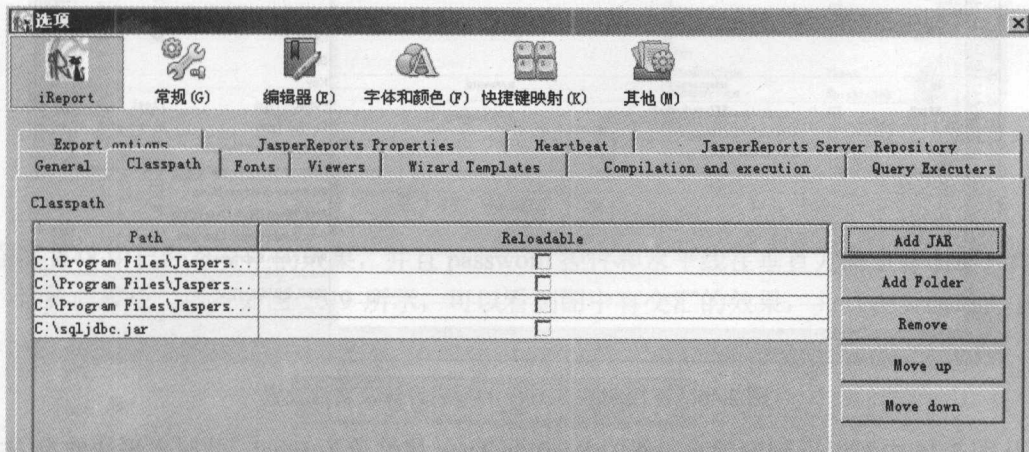


图 2.44 添加 JDBC 驱动

添加 JDBC 数据源，设置 SQL Server 数据库的连接选项如图 2.45 所示。

从图 2.49 中可以看到在第 1 页的最下方打印出了水平线。

经过此示例可以得到结论, 值为 Float 时, 控件 A 必须和控件 B 为垂直关系时才有效, 如果在水平线上方无高度自动增加的控件, 则 Float 值和 Fix Relative to Top 值效果相同。

2.3.4 Stretch Type 属性

此属性的功能是: 定义当 Band 高度变化时, 其中的控件高度是如何计算的, 初始报表界面的设计如图 2.50 所示。

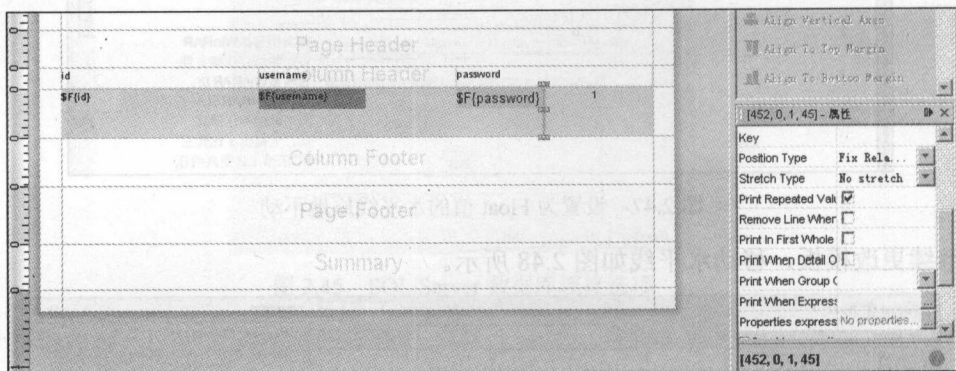


图 2.50 初始报表界面

1. No stretch

在初始报表界面中垂直线的 Stretch Type 值为默认 No stretch, 在这里需要留意的是, 垂直线并不是与 Detail 1 的 Band 同高, 之间还有 1px 的距离, 垂直线与 1px 比较的放大效果如图 2.51 所示。

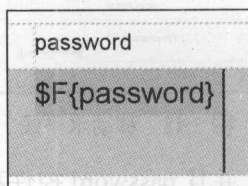


图 2.51 垂直线与 1px 的放大效果

运行效果如图 2.52 所示。

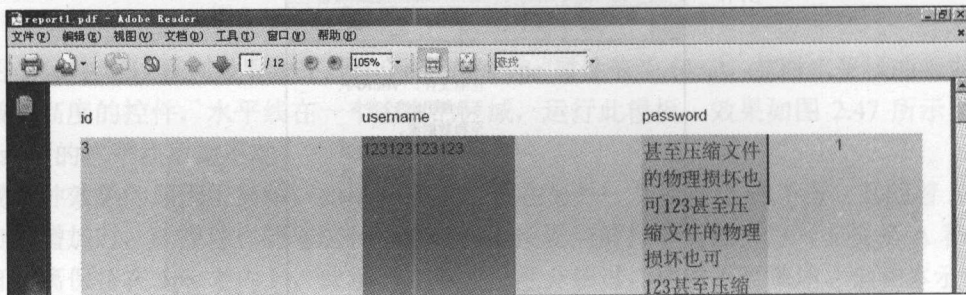


图 2.52 运行效果 1

2. Relative to Band Height

从图 2.52 中可以看到，垂直线的高度不随着 Band 高度的加大而加大，更改垂直线的属性为 **Stretch Type** **Relative to Band Height**，此属性值的含义是：控件的高度随着 Band 的高度而按比例变化，但其与底线的距离始终保持不变，在后面的示例中将利用 1px 像素来演示这个距离保持不变的效果，但测试的运行效果如图 2.53 所示。

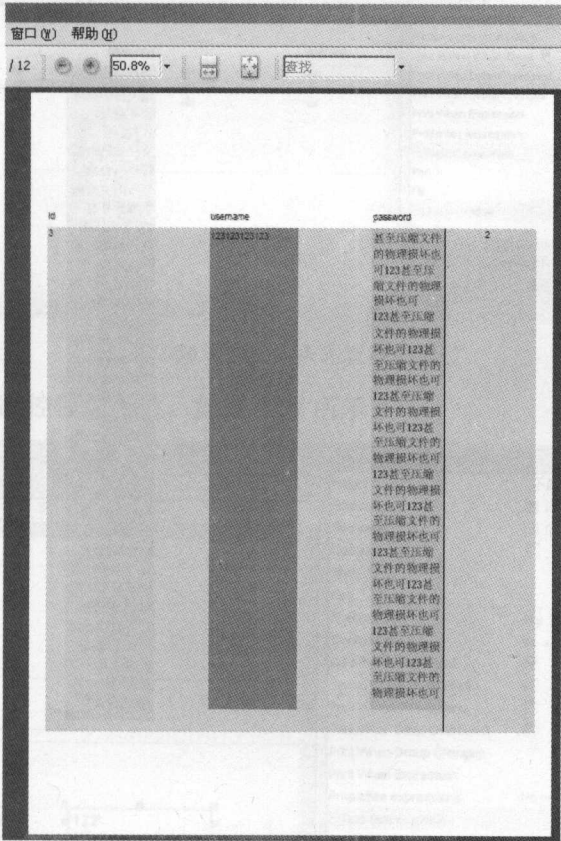


图 2.53 运行效果 2

从图 2.53 中可以看到 Band 高度变化后垂直线的高度也随着发生了变化，并且距离底线永远有距离，这点可以从 PDF 文件的放大效果中查看到，如图 2.54 所示。

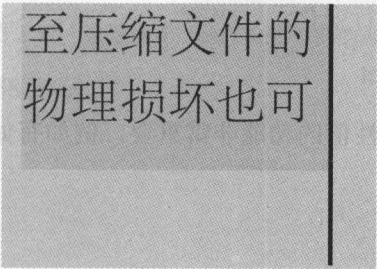


图 2.54 PDF 文件的放大效果

但垂直线仅仅出现在当前页中, 因为第 2 页并没有垂直线了, 想要实现第 2 页继续出现垂直线的效果, 将在后面的章节中介绍, 此处不再赘述。

属性值为 Relative to Band Height 时第 2 页不打印垂直线, 如图 2.55 所示。

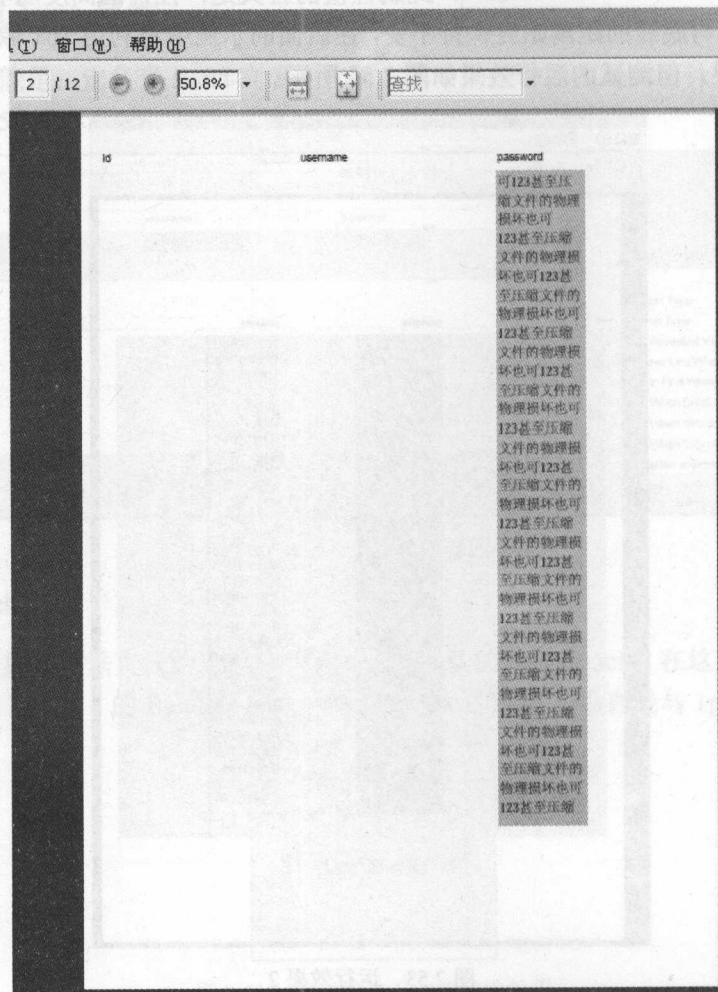


图 2.55 属性值为 Relative to Band Height 时第 2 页不打印垂直线

至此已经将属性值 Relative to Band Height 进行了效果演示, 此值的主要功能是设置控件的高度按比例与 Band 的高度进行变化。

3. Relative to Tallest Object

Relative to Tallest Object 属性值的功能非常重要, 例如有如下的报表模板, 设置 4 条垂直线的属性如图 2.56 所示。

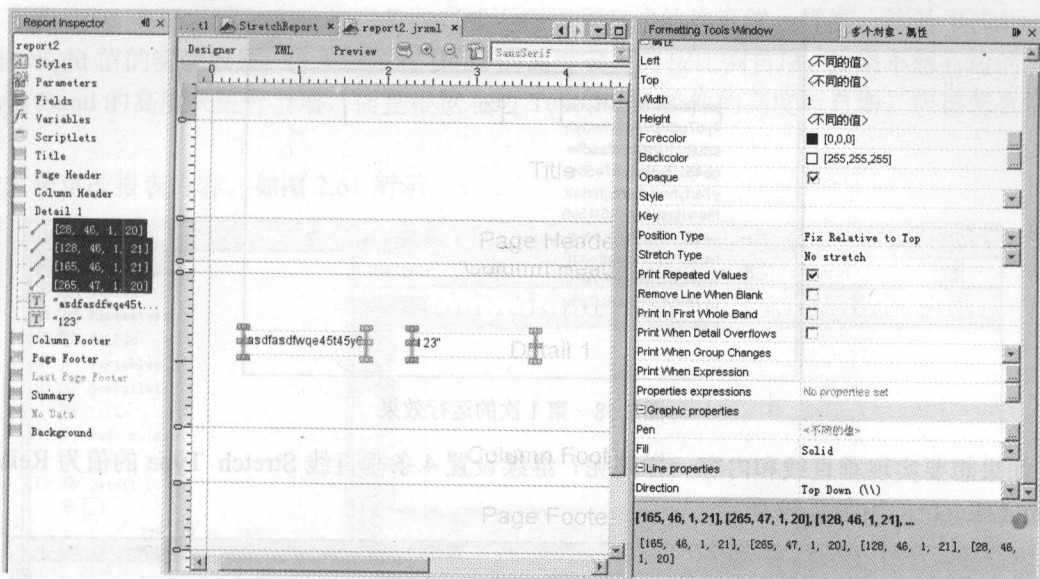


图 2.56 设置 4 条垂直线的属性

设置两个 Text Field 控件的属性，如图 2.57 所示。

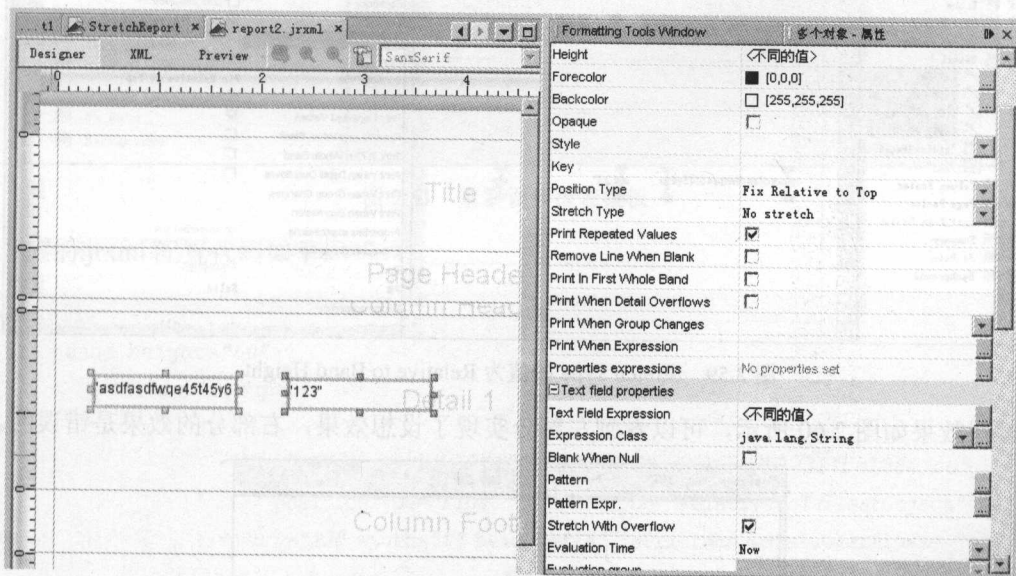


图 2.57 设置两个 Text Field 控件的属性

第 1 次的运行效果如图 2.58 所示。

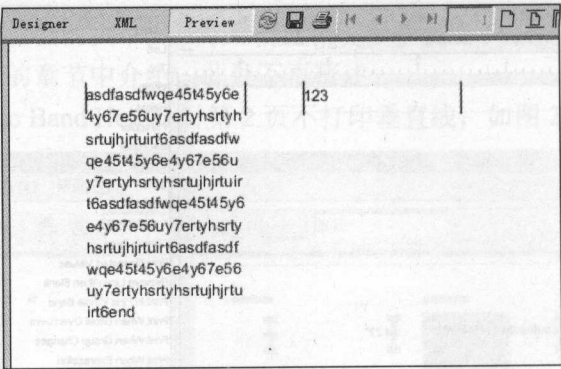


图 2.58 第 1 次的运行效果

如果想要实现垂直线和内容一样高呢？继续设置 4 条垂直线 Stretch Type 的值为 Relative to Band Height，如图 2.59 所示。

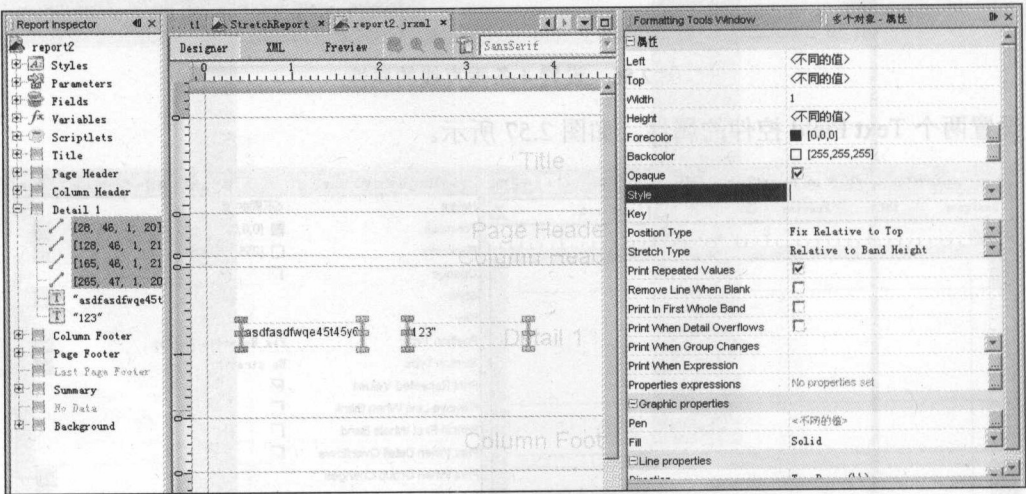


图 2.59 Stretch Type 的值为 Relative to Band Height

运行效果如图 2.60 所示，可以看到左部分实现了设想效果，右部分的效果是错误的。

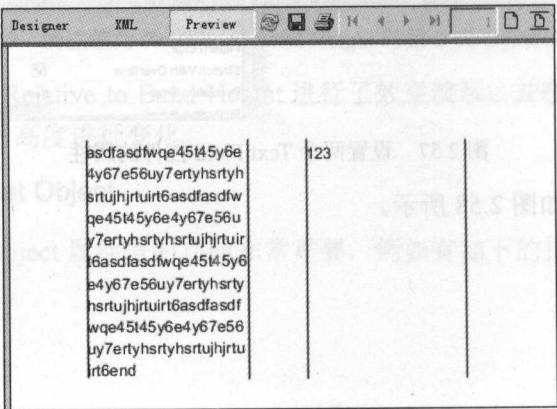


图 2.60 左右效果不相同

在图 2.60 中可以看到右边的两条垂直线不应该和左边的垂直线一样高，这是 Relative to Band Height 值的实现效果，将 4 条线和 Band 的高度设置为按比例自增，如果不想右边的垂直线根据 Band 的高度来进行自增，而是根据右边 Text Field 控件的高度而自增，应该怎么实现呢？

重新更改报表内容，如图 2.61 所示。

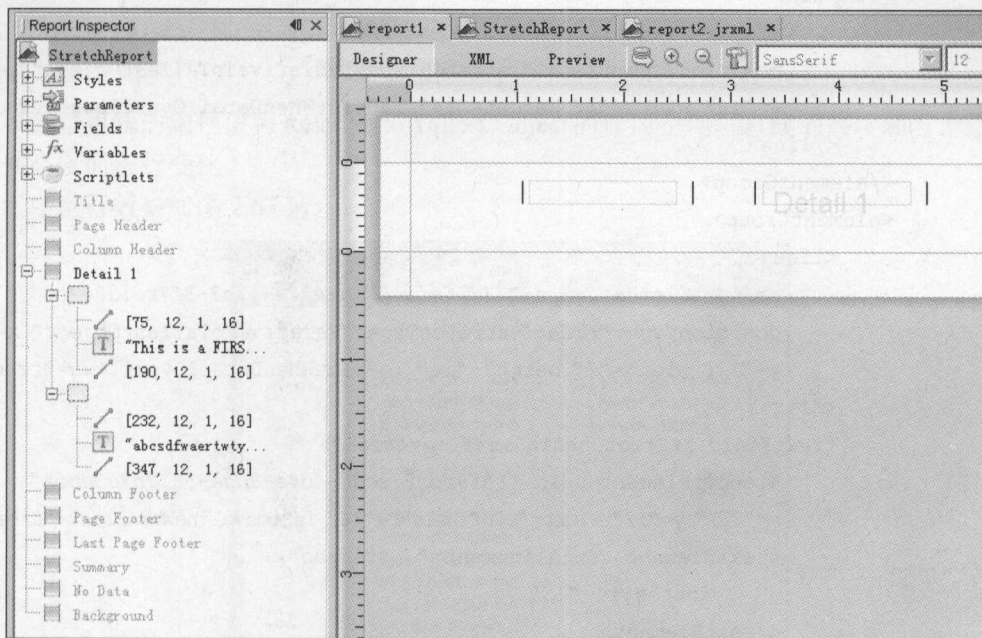


图 2.61 重新设计报表模板

对应的.jrxml 配置代码如下：

```
<detail>
  <band height="60">
    <elementGroup>
      <line>
        <reportElement uuid="71316afa-073b-4332-a7b3-3f7be13807cd"
          positionType="Float" stretchType="RelativeToTallestObject" x="75"
          y="12" width="1" height="16" isPrintWhenDetailOverflows="true" />
      </line>
      <textField isStretchWithOverflow="true">
        <reportElement uuid="8568fdc2-9c08-4d4e-8dde-5a12b2e3ebf2"
          x="80" y="12" width="100" height="16" isRemoveLineWhenBlank="true" />
        <textElement textAlignment="Justified"><font size="12" />
        </textElement>
        <textFieldExpression><![CDATA["This is a FIRST long chunk of
          text that will cause the text field to stretch outside its
          defined height and force other elements to move downwards.This
```

```

        is a FIRST long chunk of text that will cause the text field
        to stretch outside its defined height and force other elements to move
        downwards.ENDENDEND"]]]>
    < /textFieldExpression>
</textField>
<line>
    <reportElement uuid="93400e5c-fb37-4890-b514-4b8c32291bbd"
        positionType="Float" stretchType="RelativeToTallestObject" x="190"
        y="12" width="1" height="16" isPrintWhenDetailOverflows="true" />
</line>
</elementGroup>
<elementGroup>
    <line>
        <reportElement uuid="71316afa-073b-4332-a7b3-3f7be13807cd"
            positionType="Float" stretchType="RelativeToTallestObject" x="232"
            y="12" width="1" height="16" isPrintWhenDetailOverflows="true" />
    </line>
    <textField isStretchWithOverflow="true">
        <reportElement uuid="8568fdc2-9c08-4d4e-8dde-5a12b2e3ebf2"
            x="237" y="12" width="100" height="16" isRemoveLineWhenBlank="true" />
        <textElement textAlignment="Justified">
            <font size="12" />
        </textElement>
        <textFieldExpression>
            <![CDATA["abcsdfwaertwtyweryertyhdtudrtuujdrusrtuyuirftitdy"]]>
        </textFieldExpression>
    </textField>
    <line>
        <reportElement uuid="93400e5c-fb37-4890-b514-4b8c32291bbd"
            positionType="Float" stretchType="RelativeToTallestObject" x="347"
            y="12" width="1" height="16" isPrintWhenDetailOverflows="true" />
    </line>
</elementGroup>
</band>
</detail>

```

在上面的代码中使用了新的标签<elementGroup>，它的作用可以使控件进行分组，并且设置 Stretch Type 为 Relative to Tallest Object，如图 2.62 所示。

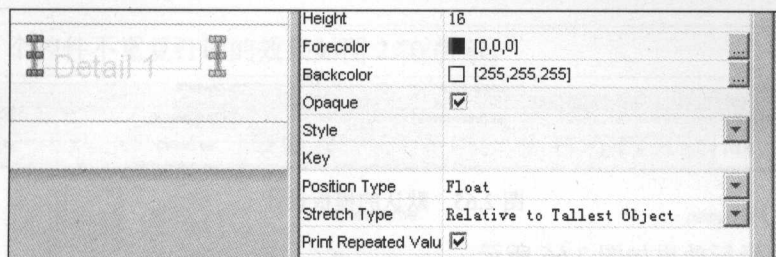


图 2.62 选择 Relative to Tallest Object

设置垂直线的属性值为 Relative to Tallest Object 的作用是使垂直线的高度随着本组内最高的控件高度而进行变化。

程序运行效果如图 2.63 所示。

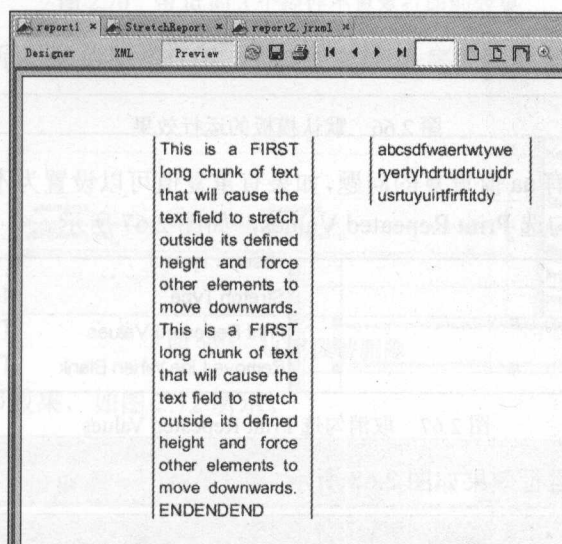


图 2.63 最终运行效果

2.3.5 Print Repeated Values 属性

在 MySQL 中创建数据表结构及数据内容，userinfo 表中的数据如图 2.64 所示。

Filter:			
	id	username	password
▶	1	a	aa
	2	a	aa
	3	b	bb
	4	c	cc
*	NULL	NULL	NULL

图 2.64 userinfo 表中的数据

报表模板的默认设置如图 2.65 所示。

Page Header		
id	username	password
\$F{id}	\$F{username}	\$F{password}

图 2.65 默认模板设计

默认模板的运行效果如图 2.66 所示。

id	username	password
1	a	aa
2	a	aa
3	b	bb
4	c	cc

图 2.66 默认模板的运行效果

可以发现 password 有 aa 值重复的问题,如果有重复值可以设置为不打印,即改变 password 控件的属性设置,取消勾选 Print Repeated Values, 如图 2.67 所示。

password	
\$F{password}	

Position Type	Fix...
Stretch Type	No ...
Print Repeated Values	<input type="checkbox"/>
Remove Line When Blank	<input type="checkbox"/>

图 2.67 取消勾选 Print Repeated Values

取消勾选的属性值运行效果如图 2.68 所示。

id	username	password
1	a	aa
2	a	
3	b	bb
4	c	cc

图 2.68 取消勾选的属性值运行效果

以上是单独对 password 设置了属性,如果这 3 个控件都设置为不打印重复的数据会是什么效果呢?即 3 个控件都设置为不打印重复值,如图 2.69 所示。

id	username	password
\$F{id}	\$F{username}	\$F{password}

Position Type	Fix...
Stretch Type	No ...
Print Repeated Values	<input type="checkbox"/>
Remove Line When Blank	<input type="checkbox"/>

图 2.69 3 个控件都设置为不打印重复值

带 id 的 3 个控件不重复打印的效果如图 2.70 所示。

id	username	password
1	a	aa
2	b	bb
3	c	cc

图 2.70 带 id 的 3 个控件不重复打印的效果

由于 id 不重复，所以 id 值照常打印，那如果去掉 id 字段呢？删除 id 字段的报表设计如图 2.71 所示。

Page Header	Column Header	password
username	detail 1	\$F(password)
Column Footer		

图 2.71 id 字段被删除

去掉 id 字段的打印效果，如图 2.72 所示。

username	password
a	aa
b	bb
c	cc

图 2.72 去掉 id 字段的打印效果

2.3.6 Remove line when blank 属性

在图 2.72 中可以看到打印了一个空行，并且第 2 行和第 3 行之间有间距，如果想把空行删除并且去掉行距怎么办？很简单，使用 Remove line when blank 属性即可实现。

在设置此属性之前先把报表模板改动一下，将 Text Field 控件的高度设置为和 Band 一样高，这样多行之间就没有了间距，如图 2.73 所示。

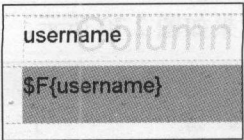


图 2.73 Text Field 高度和 Band 一样高

设置 username 和 password 属性，如图 2.74 所示。

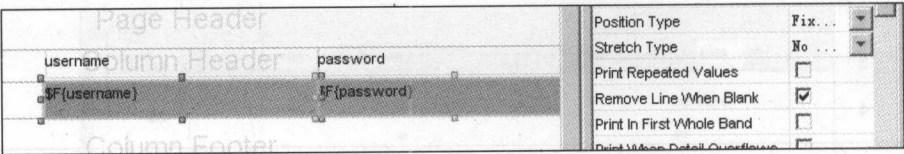


图 2.74 设置属性值

删除了空行并且无间距的效果如图 2.75 所示。

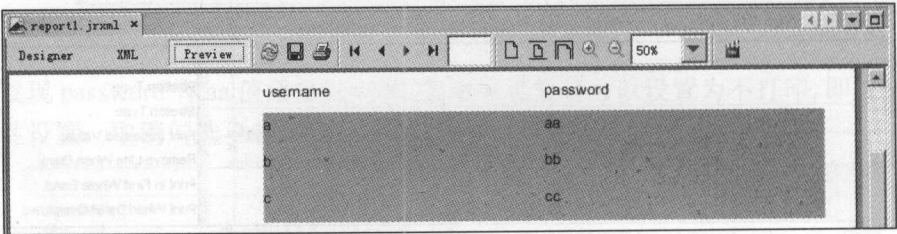


图 2.75 删除了空行并且无间距的效果

2.3.7 Print In First Whole Band 属性

在上面的示例中数据表 userinfo 记录较少，而且重复的字段值也比较少，添加 userinfo 数据表中的记录如图 2.76 所示。

Filter: [] Edit: [] Export: []		
	id	username
1	中国中国中国	abcxyz123456
2	a	aa
3	b	bb
4	c	cc
5	a	aaa
6	a	aaa
7	a	aaa
8	a	aaa
9	a	aaa
10	a	aaa
11	a	aaa
12	a	aaa
13	a	aaa
14	a	aaa

图 2.76 添加 userinfo 数据表中的记录

从图 2.76 中可以看到重复的记录较多，设置 username 和 password 属性，如图 2.77 所示。

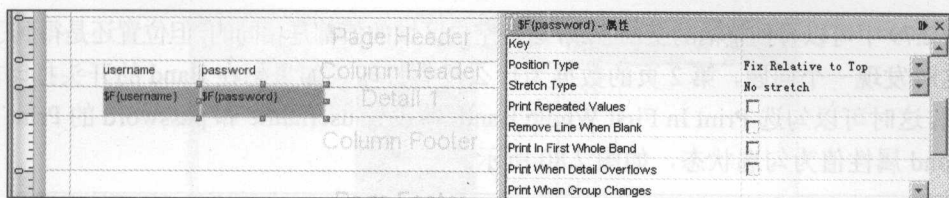


图 2.77 设置 username 和 password 属性值

运行效果如图 2.78 所示。

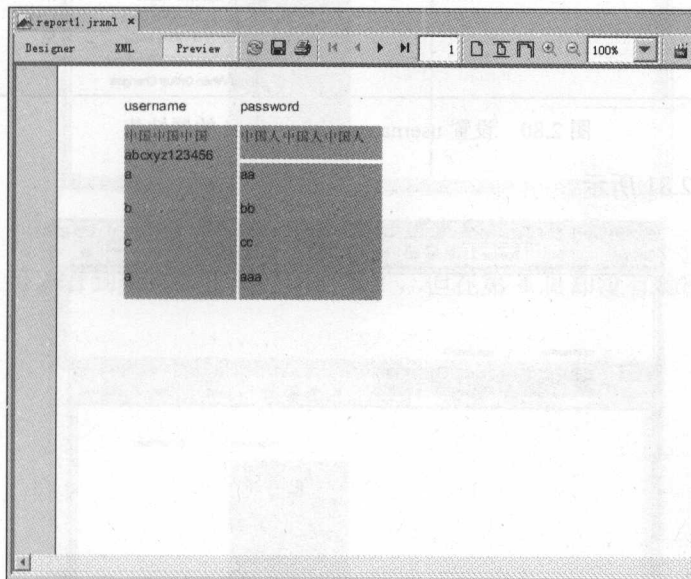


图 2.78 运行效果

由于在前面将 username 和 password 设置为不打印重复的数据，所以整个第 1 页的空白区域较多，再来看看第 2 页的运行效果，如图 2.79 所示。

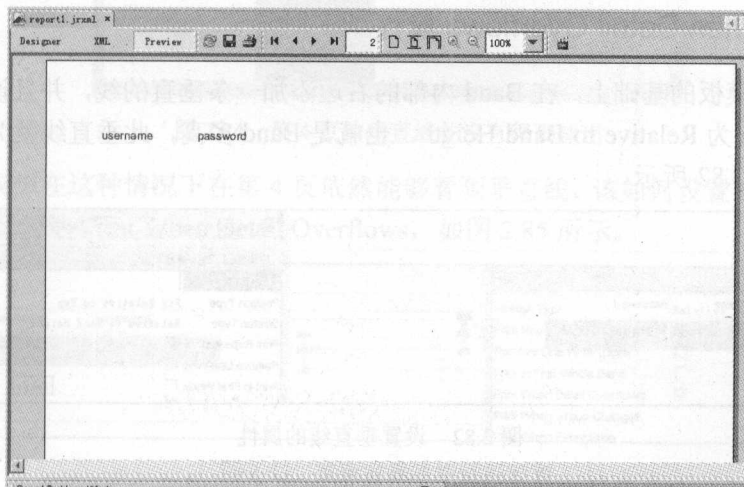


图 2.79 运行效果 1 的第 2 页

从图 2.79 中可以看到整幅的空白区域,这些空白区域的值都是相同的,但位置还是得到了保留。

但这时发现一个问题,第 2 页的数据为什么没有打印? 如果想在 Band 的开头打印前面重复的数据,这时可以勾选 Print In First Whole Band,即设置 username 和 password 的 Print In First Whole Band 属性值为勾选状态,如图 2.80 所示。

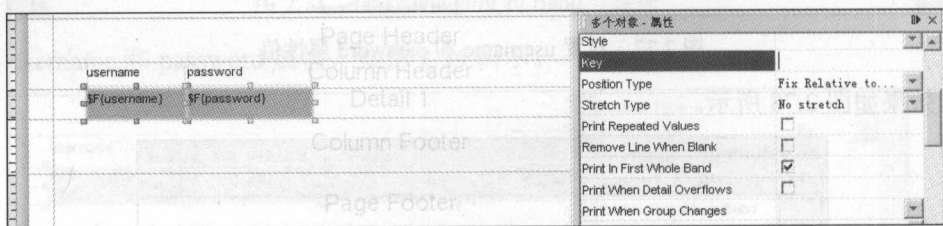


图 2.80 设置 username 和 password 的属性值

运行效果如图 2.81 所示。

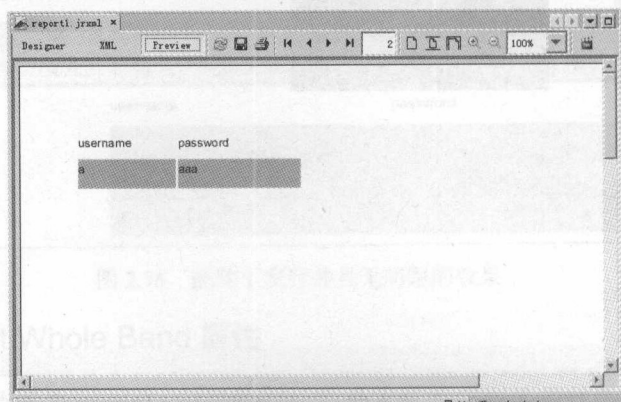


图 2.81 运行效果

在图 2.81 中看到在第 2 页的开始打印出了前面重复的数据值。

2.3.8 Print When Detail Overflows 属性

在前面报表模板的基础上,在 Band 内部的右边添加一条垂直的线,并且设置此垂直线的属性 Stretch Type 为 Relative to Band Height,也就是 Band 多高,此垂直线就为多高,设置垂直线的属性如图 2.82 所示。

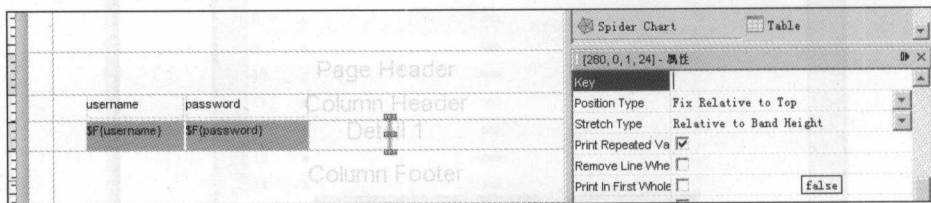


图 2.82 设置垂直线的属性

此报表运行后,在字段较多的页面的打印效果如图 2.83 所示。

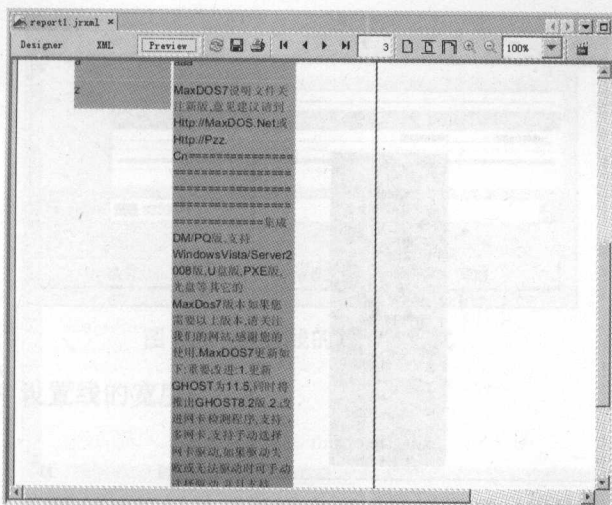


图 2.83 字符较多时的打印效果

通过图 2.83 可以看到垂直线正确打印出来了，但在第 4 页却没有输出垂直线，效果如图 2.84 所示。

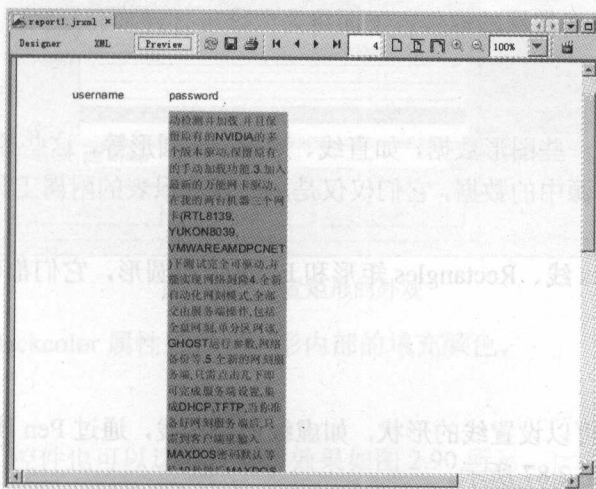


图 2.84 第 4 页的垂直线并没有得到输出

那么，如果想在這種情況下在第 4 页依然能够看到垂直线，该如何设置呢？重新对垂直线进行属性设置，勾选 Print When Detail Overflows，如图 2.85 所示。

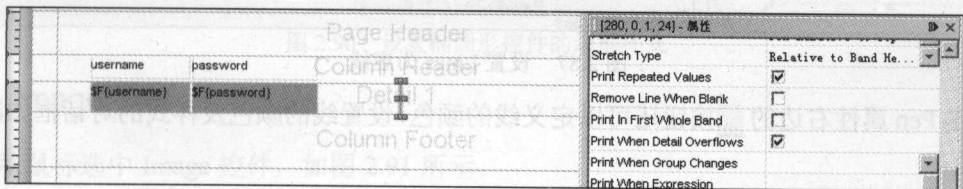


图 2.85 重新对垂直线进行设置

再次运行报表，在第 4 页终于又出现垂直线了，效果如图 2.86 所示。

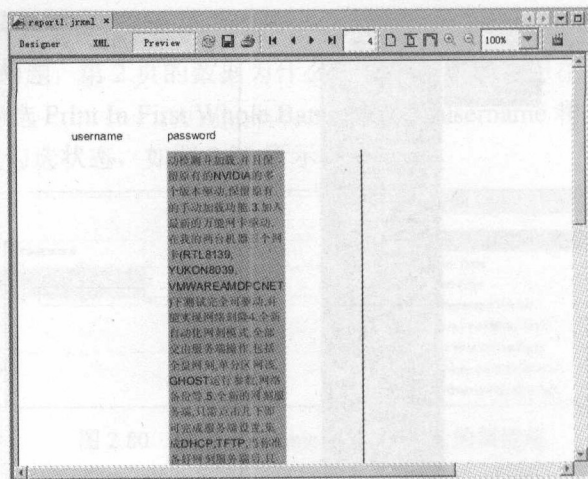


图 2.86 第 4 页显示垂直线

2.4 控件的使用方法

iReport 工具中有很多常用的控件, 下面将为大家进行详细介绍。

2.4.1 形状控件

图形控件用于显示一些图形数据, 如直线、矩形、椭圆形等, 这些控件有一个比较重要的特点就是不能显示数据源中的数据, 它们仅仅是用于设计报表的附属工具, 从而使设计的报表更加美观、大方、得体。

图形控件包含 Lines 线、Rectangles 矩形和 Ellipses 椭圆形, 它们都可以设置边框的样式, 以及填充的内容。

1. Lines

针对 Lines 控件, 可以设置线的形状, 如虚线还是实线, 通过 Pen 属性来设置外观, 设置 Lines 为虚线的界面如图 2.87 所示。

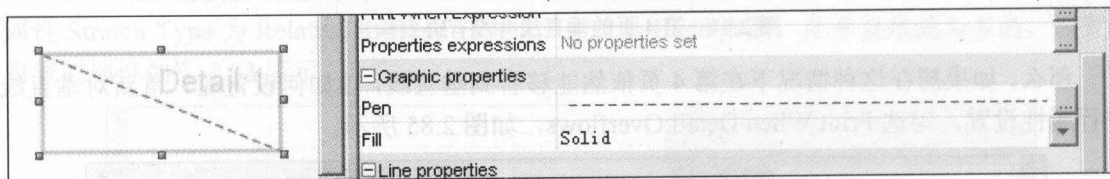


图 2.87 设置 Lines 为虚线

单击 Pen 属性右边的 ... 按钮还可以定义线的颜色, 设置线的颜色及样式的对话框如图 2.88 所示。

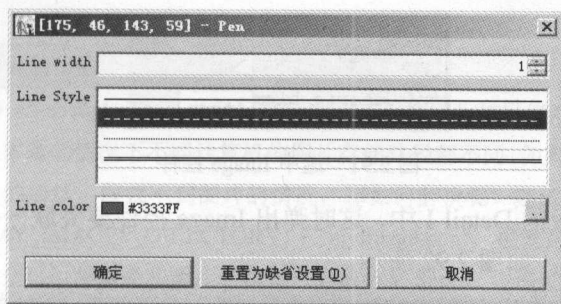


图 2.88 设置线的颜色及样式

还可以在图 2.88 中设置线的宽度。

2. Rectangles

Rectangles（矩形）控件的边框线外观同样也可以这样设置，设置界面如图 2.89 所示。

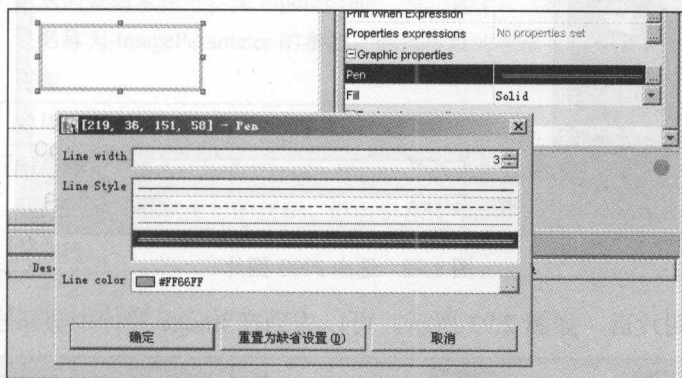


图 2.89 设置矩形的外观

还可以通过设置 Backcolor 属性来改变矩形内部的填充颜色。

3. Ellipses

Ellipses（椭圆形）控件也可以这样设置，效果如图 2.90 所示。



图 2.90 设置椭圆形控件的边框样式

2.4.2 Image 控件

利用鼠标选中 Image 控件，如图 2.91 所示。

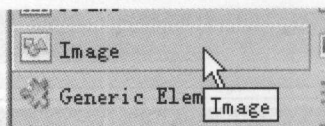


图 2.91 选中 Image 控件

然后拖曳到报表模板的 Detail 1 中，这时弹出 Image 控件要关联一个真正的图片文件选择对话框，选中 PNG 图片，如图 2.92 所示。

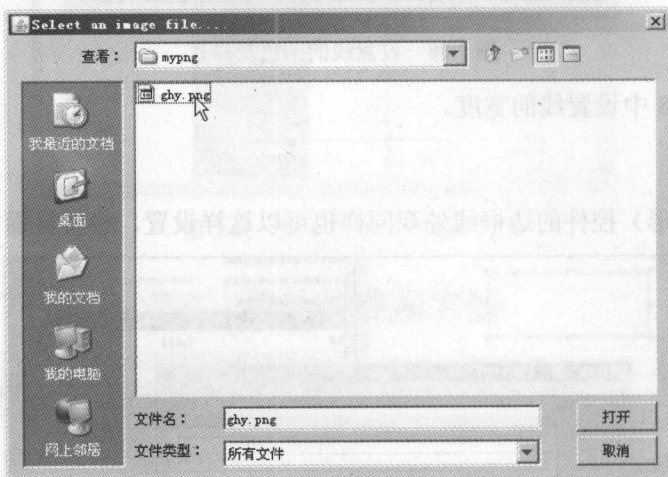


图 2.92 选中 PNG 图片

选中要显示的图片后，如图 2.93 所示，可以看到在 Image 控件中显示出了 PNG 图片。

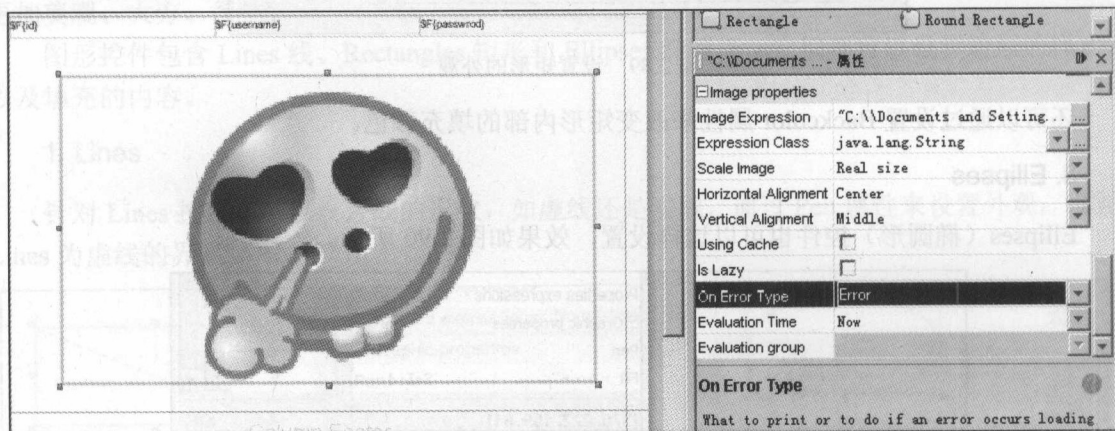


图 2.93 在 Image 控件中显示出了 PNG 图片

选中 Image 控件后需要留意右边的 Image Expression 属性，此时的值为：“C:\\Documents and Settings\\ghy\\桌面\\mypng\\ghy.png”，说明此 Image 控件使用的是绝对路径，注意文件夹的“\\”要转义变成“\\\\”。

需要说明的是 Image Expression 属性共支持 6 种值类型，详细说明如表 2.1 所示。

表 2.1 6 种值类型

类型	具体值的写法
java.lang.String	<p>(1) c:\abc\ghy.jpg, 此种写法不建议使用, 因为使用绝对路径不便于项目的移植, 在大多数的情况下不同计算机的图片存储路径都不同, 如果使用绝对路径将会造成非常大的麻烦, 即找不到图片的情况, 那如何解决呢? 建议使用如下参数式的写法: <code>\$P{MY_IMAGES_DIRECTORY} + "myImage.png"</code>, 也就是将路径写成一个参数, 或者完全可以将路径和图片名作为一个参数从 Servlet 传递过来, 写法为 <code>\$P{ImagePath}</code></p> <p>(2) com/gaohongyan/www/ghy.png</p> <p>(3) http://localhost:8080/reportShow/ghy.png</p>
java.io.File	可以使用 Java 代码来加载图片: “new java.io.File("c:\ghy.jpg")”
java.net.URL	可以使用代码访问远程 URL 中的图片资源: “new java.net.URL("http://127.0.0.1/test.jpg")”
java.io.InputStream	报表的数据来源可以是 InputStream, 写法如下: “\$P{imageParameter}”, 表示来源是名称为 imageParameter 的参数, 但该参数的数据类型是 InputStream 中得到图片的资源
java.awt.Image	图片来自于 java.awt.Image 对象
JRRenderable	图片来自于 net.sf.jasperreports.engine.JRRenderable 接口, 使用此接口的目的是程序员自己实现绘制图形的代码, 但由于在使用率上较少, 所以具体使用方法请参考官方文档

下面将分别介绍前 5 种写法的具体使用。

1. java.lang.String

在报表模板中设计 Image 控件的 Image Expression 属性值为如下 XML 代码:

```
<image scaleImage="RealSize" hAlign="Center" vAlign="Middle" isUsingCache="false">
  <reportElement uuid="166f5330-0a26-4370-95b3-3e1f8a366b8b"
    x="340" y="28" width="146" height="118" />
  <imageExpression><![CDATA["com/gaohongyan/www/ghy.png"]]></imageExpression>
</image>
<image>
  <reportElement uuid="ec75d273-69e8-4311-b8df-f9e68198b4d3"
    x="340" y="172" width="147" height="122" />
  <imageExpression><![CDATA["http://localhost:8081/reportShow/ghy.png"]]>
  </imageExpression>
</image>
```

上面的两种写法很明显是要报表运行在 Web 项目中, 并且其中一个 Image 控件要从 src 资源路径的包中取得 PNG 图片资源, java.lang.String 示例的项目结构如图 2.94 所示。

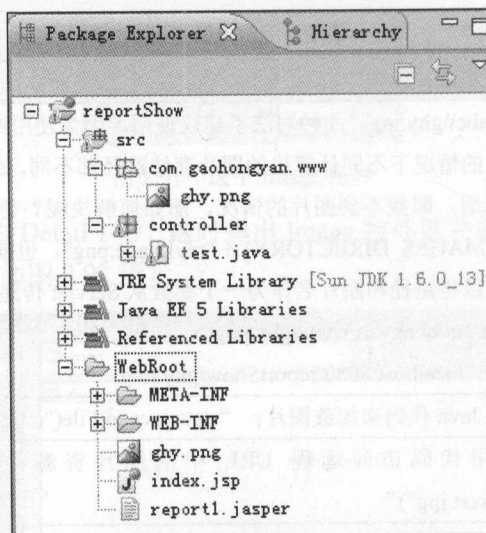


图 2.94 java.lang.String 示例的项目结构

名称为 test 的 Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext().
                getResourceAsStream("report1.jasper");
            JasperRunManager.runReportToPdfStream(reportStream,
                servletOutputStream, new HashMap(), new JREmptyDataSource());
            response.setContentType("application/pdf");
            servletOutputStream.flush();
            servletOutputStream.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

java.lang.String 的运行效果如图 2.95 所示。

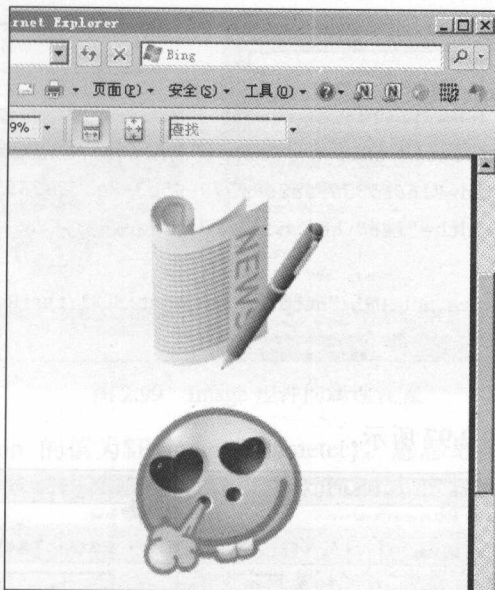


图 2.95 java.lang.String 运行效果

2. java.io.File

在报表模板中设计 Image 属性的 XML 代码如下：

```
<image scaleImage="RealSize" hAlign="Center" vAlign="Middle" isUsingCache="false">
  <reportElement uuid="166f5330-0a26-4370-95b3-3e1f8a366b8b"
    x="255" y="90" width="146" height="118" />
  <imageExpression><![CDATA[new java.io.File("c:\\ghyl23.png")]]>
  </imageExpression>
</image>
```

java.io.File 的运行效果如图 2.96 所示。

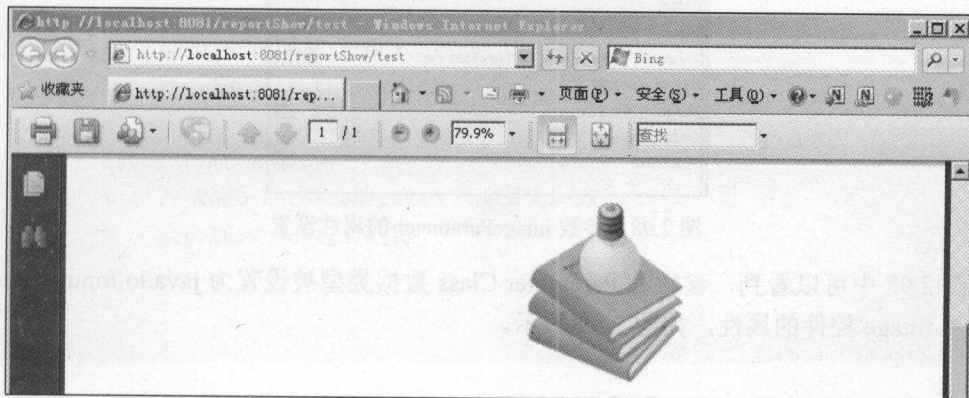


图 2.96 java.io.File 运行效果

3. java.net.URL

在报表模板中设计 Image 属性的 XML 代码如下：

```
<image scaleImage="RealSize" hAlign="Center" vAlign="Middle" isUsingCache="false">
  <reportElement uuid="166f5330-0a26-4370-95b3-3e1f8a366b8b"
    x="220" y="101" width="146" height="118" />
  <imageExpression>
    <![CDATA[new java.net.URL("http://localhost:8081/testReportRemotePNG/123.png")]]>
  </imageExpression>
</image>
```

报表运行后的效果如图 2.97 所示。

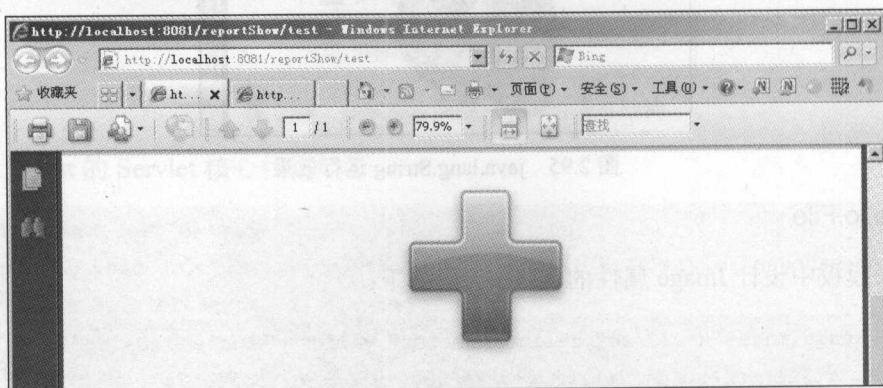



图 2.97 运行效果

4. java.io.InputStream

在报表中添加一个名称为  imageParameter 的参数，它的数据类型设置如图 2.98 所示。

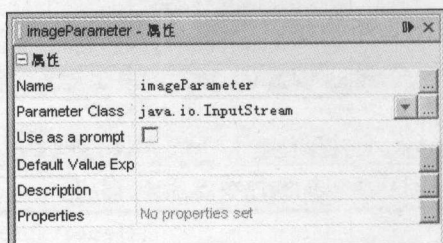


图 2.98 参数 imageParameter 的属性设置

从图 2.98 中可以看到，参数的 Parameter Class 数据类型被设置为 java.io.InputStream 类。继续设置 Image 控件的属性，如图 2.99 所示。

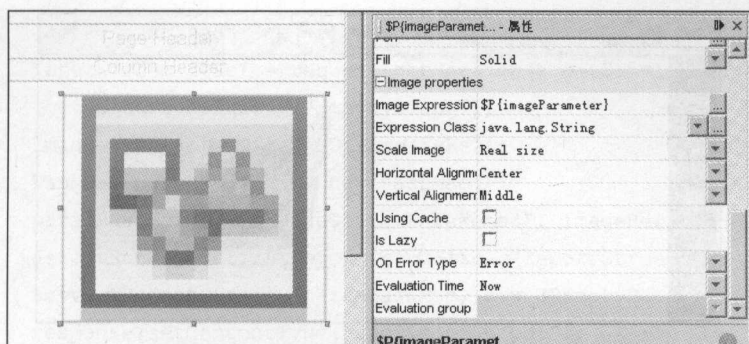


图 2.99 Image 控件的属性设置

属性 Image Expression 的值为 `$P{imageParameter}`，意思是从参数 `imageParameter` 取得 `InputStream` 图片输入流，然后以图片方式显示出来。

核心的 Servlet 代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report1.jasper");
            InputStream isRef = request.getSession().getServletContext()
                .getResourceAsStream("/ghy.png");
            HashMap paramHashMap = new HashMap();
            paramHashMap.put("imageParameter", isRef);
            JasperRunManager.runReportToPdfStream(reportStream,
                servletOutputStream, paramHashMap, new JREmptyDataSource());
            response.setContentType("application/pdf");
            servletOutputStream.flush();
            servletOutputStream.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

`InputStream` 输入流显示成图片后的效果如图 2.100 所示。

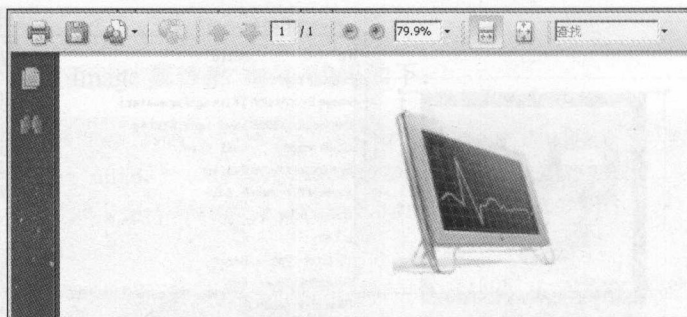


图 2.100 InputStream 输入流显示成图片

5. java.awt.Image

通过使用 `java.awt.Image` 对象可以自定义打印的内容，但在本示例中还是使用 PNG 图片资源转成 `Image` 对象的方法来打印图形。

在报表中添加参数 `imageObjectParameter`，并且设置该参数的属性，如图 2.101 所示。

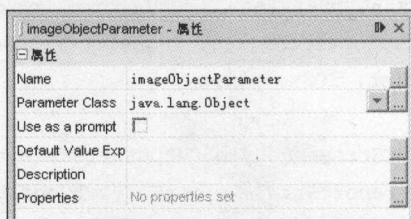


图 2.101 设置 imageObjectParameter 的参数属性

设置 `Image` 控件的属性，如图 2.102 所示。

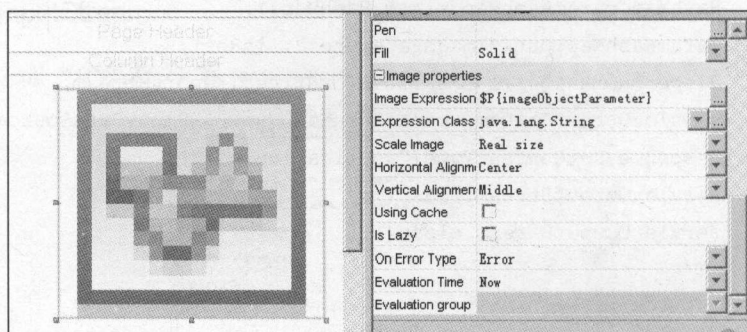


图 2.102 设置 Image 控件的属性

服务器端的 Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
```



```

        .getResourceAsStream("report1.jasper");
        InputStream isRef = request.getSession().getServletContext()
            .getResourceAsStream("/ghyxyz.png");
        Image imageRef = ImageIO.read(isRef);
        HashMap paramHashMap = new HashMap();
        paramHashMap.put("imageObjectParameter", imageRef);
        JasperRunManager.runReportToPdfStream(reportStream,
            servletOutputStream, paramHashMap, new JREmptyDataSource());
        response.setContentType("application/pdf");
        servletOutputStream.flush();
        *servletOutputStream.close();
    }
    catch (JRException e)
    { // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

打印图片来源为 Image 对象的效果如图 2.103 所示。

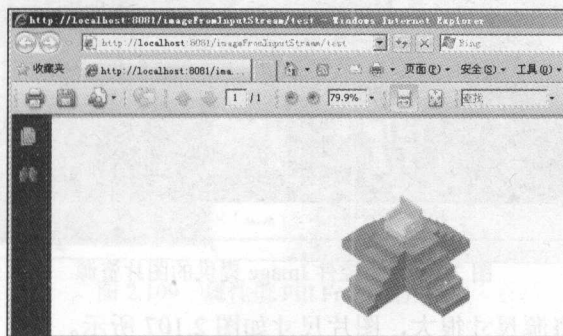


图 2.103 打印图片来源为 Image 对象的效果

2.4.3 Image 控件

控件有很多自己独有的属性，Image 控件的属性列表如图 2.104 所示。

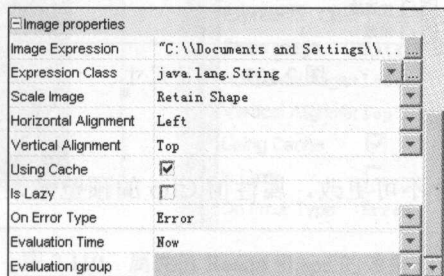


图 2.104 Image 控件属性

下面将为大家详细介绍主要属性的功能。

1. Scale Image 属性

Scale Image 属性的作用主要是设置图片的缩放方式，具有如下选项可供选择，如图 2.105 所示。

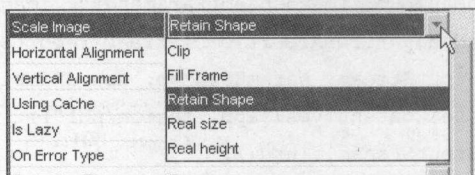


图 2.105 Scale Image 属性的可取值列表

本示例为控件 Image 提供的图片资源如图 2.106 所示。

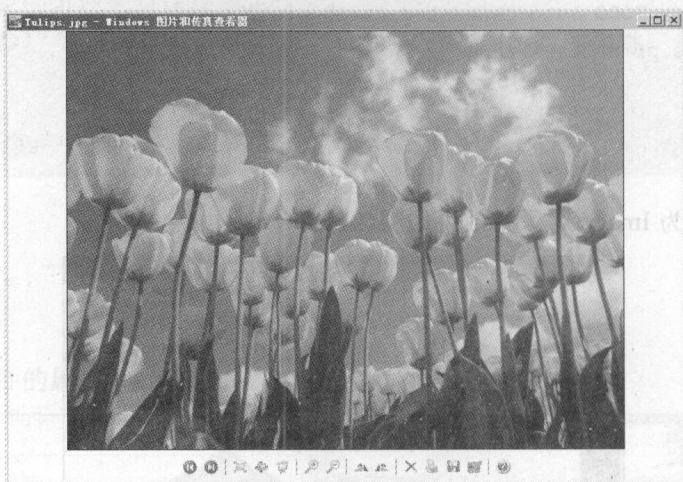


图 2.106 为控件 Image 提供的图片资源

图 2.106 中的图片资源尺寸很大，图片尺寸如图 2.107 所示。

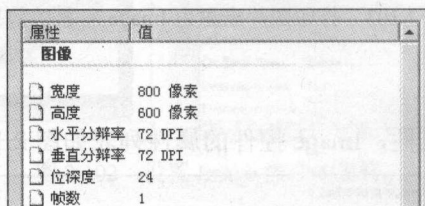


图 2.107 图片尺寸

(1) Clip

Clip 的作用是图片的尺寸不可更改，属性值 Clip 的预览效果如图 2.108 所示。

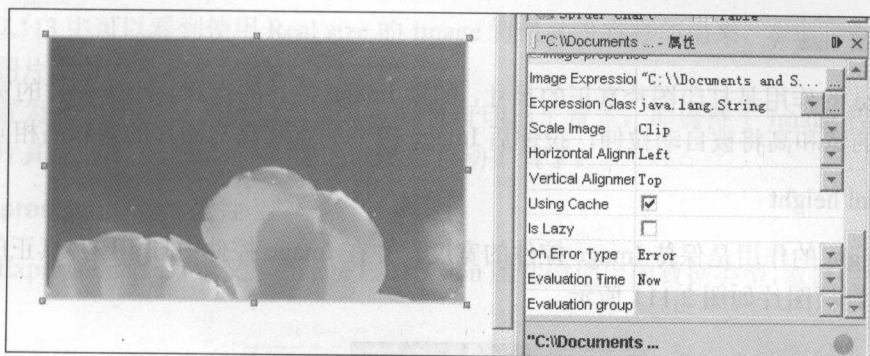


图 2.108 属性值 Clip 的效果

(2) Fill Frame

Fill Frame 的作用是图片显示的大小随着 Image 控件而改变，容易使图片产生变形，属性值 Fill Frame 的预览效果如图 2.109 所示。

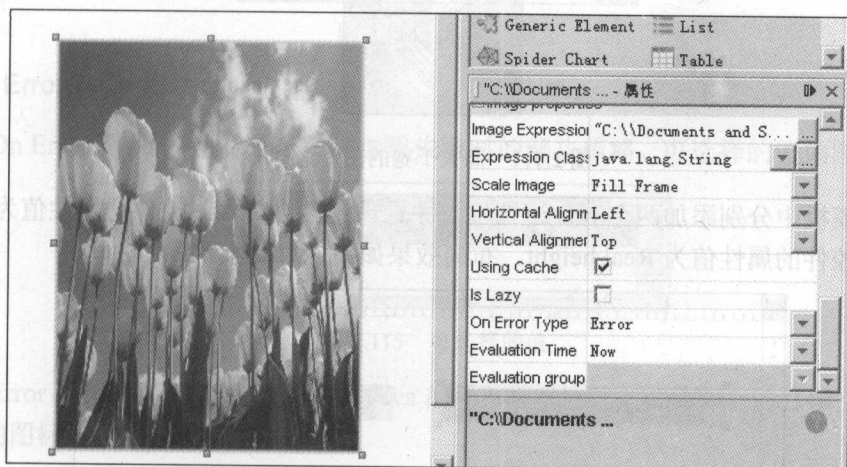


图 2.109 属性值 Fill Frame 的效果

(3) Retain Shape

Retain Shape 的作用是图片显示的大小随着 Image 控件而改变，但一直是保持图片的等比属性，属性值 Retain Shape 的预览效果如图 2.110 所示。

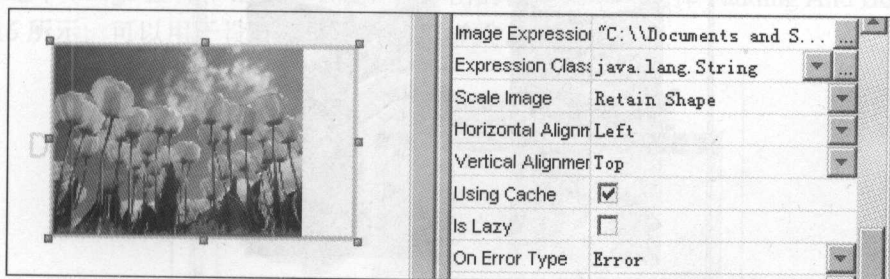


图 2.110 属性值 Retain Shape 的效果

(4) Real size

Real size 的作用是打印图片真正的尺寸, 如果 Image 控件的宽和高小于图片的宽和高, 则 Image 控件的宽和高将被自动拉伸, 拉伸后 Image 控件的宽和高与图片的宽和高相同。

(5) Real height

Real height 的作用是保持 Image 控件的宽度, 并在 Image 控件中打印图片真正的高度。例如, 素材图片如图 2.111 所示。

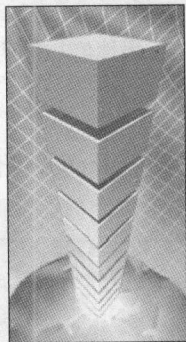


图 2.111 高大于宽的图片素材

在报表模板中分别添加两个 Image 控件, 并且设置左边 Image 控件的属性值为 Real size, 右边 Image 控件的属性值为 Real height, 布局效果如图 2.112 所示。

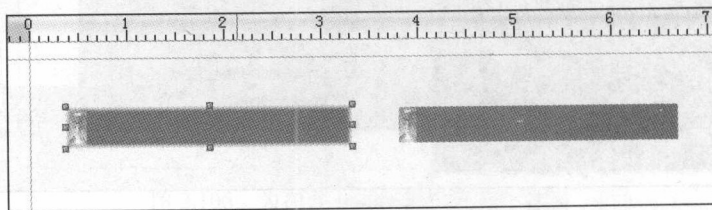


图 2.112 设计布局效果

运行效果如图 2.113 所示。

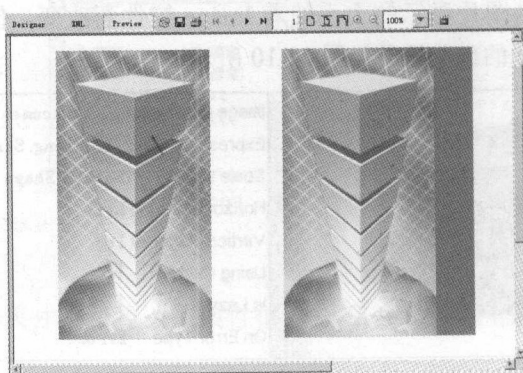


图 2.113 运行效果

从图 2.113 中可以看到使用 Real size 的 Image 背景并没有打印出来, 实现了 Image 控件的高和宽与图片素材的高和宽相同。

而设置为 Real height 属性值的 Image 控件打印出了背景, 即保持了 Image 控件的宽度, 打印了图片真正的高度, Image 控件的高度被自动扩展了。

2. Expression Class 属性

属性 Expression Class 定义了 Image Expression 属性值使用的数据类型, 取值类型如图 2.114 所示。

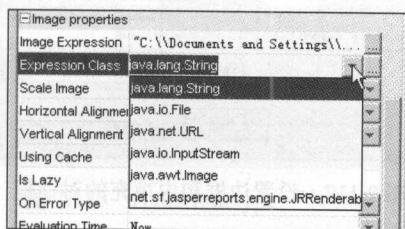


图 2.114 取值类型

3. On Error Type 属性

属性 On Error Type 的功能是当图片加载失败时的解决策略, 可选择的值如图 2.115 所示。

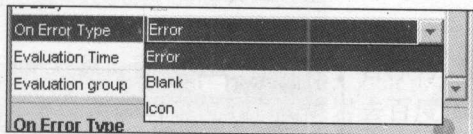


图 2.115 可选择的值

其中 Error 是指图片加载失败时报出 Java 的异常, Blank 代表出现空的打印空间, Icon 代表将默认的图标显示出来。

4. Vertical Alignment 和 Horizontal Alignment 属性

这两个属性用于定义图片在 Image 控件中的对齐位置。

5. 设置 Image 的边框及超链接

可以选中 Image 控件单击鼠标右键, 在弹出的快捷菜单中选择 Padding And Borders 命令, 如图 2.116 所示, 可以用于设置内填充、边框样式、超链接等。

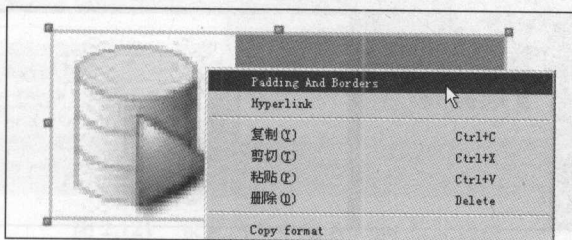


图 2.116 快捷菜单

设置边框和内填充的对话框如图 2.117 所示。

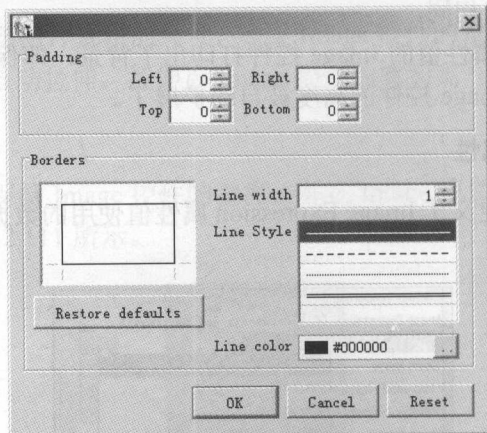


图 2.117 设置边框和内填充的对话框

设置图片的超链接也很简单，如图 2.118 所示。

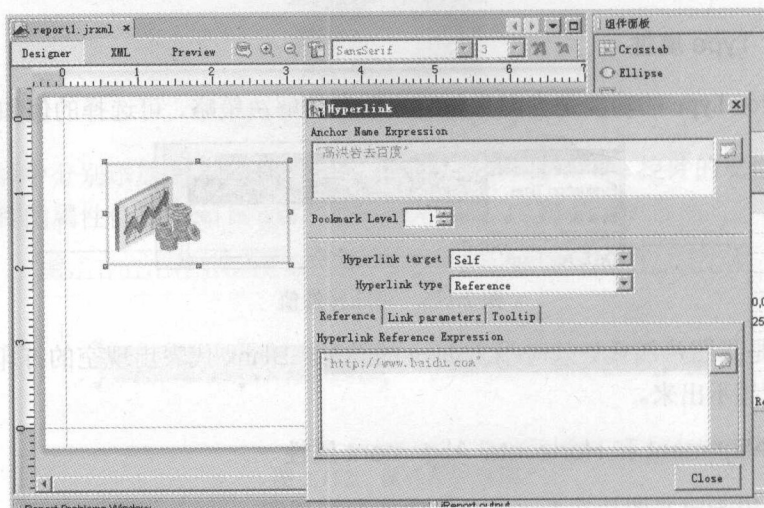


图 2.118 设置图片的超链接

程序运行后的效果（运行效果）如图 2.119 所示。

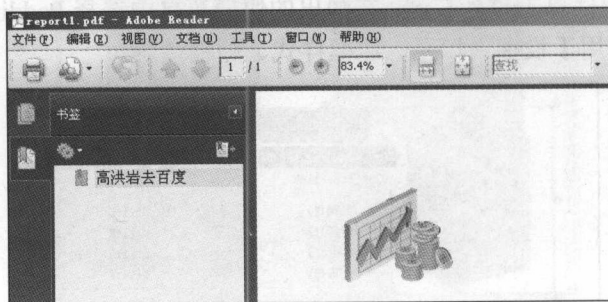


图 2.119 PDF 运行效果

在图 2.118 中属性 Hyperlink type 比较重要, 它的取值范围如图 2.120 所示。

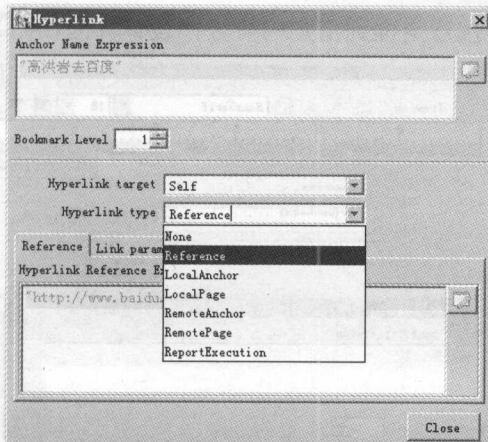


图 2.120 Hyperlink type 下拉列表

(1) Reference

通常情况下它是 URL 的唯一标识, 前面的示例就是使用此属性值定义了一个链接, 并且关联到 <http://www.baidu.com>, 而且还定义了一个名称为“高洪岩去百度”的 Anchor。

(2) LocalAnchor

定位到本地的一个 Anchor 对象, 前面已经有了一个名称为“高洪岩去百度”的 Anchor, 本示例要设置一个文本链接, 单击链接后定位到“高洪岩去百度”这个 Anchor, 添加一个 Text Field 控件, 并设置两个本地 Anchor 彼此切换, 如图 2.121 所示。

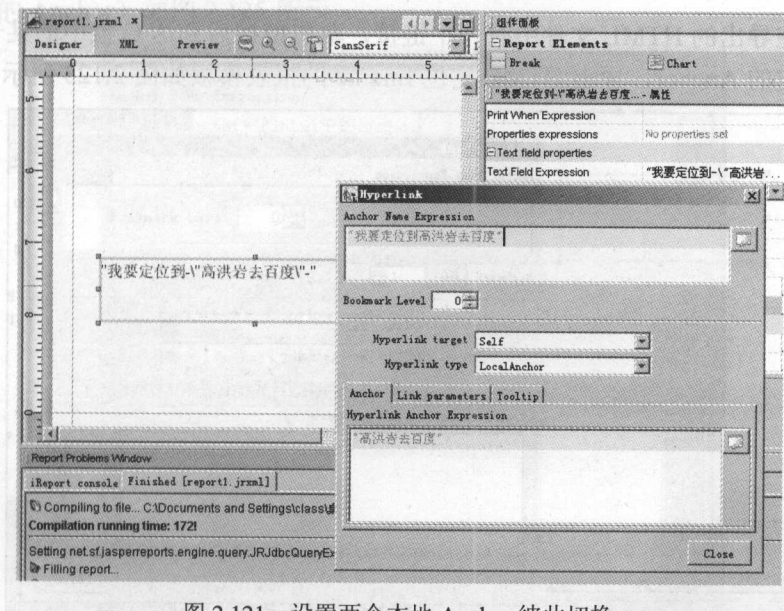


图 2.121 设置两个本地 Anchor 彼此切换

(3) LocalPage

该属性值可以在单击链接后定位到指定的页数，设计界面如图 2.122 所示。

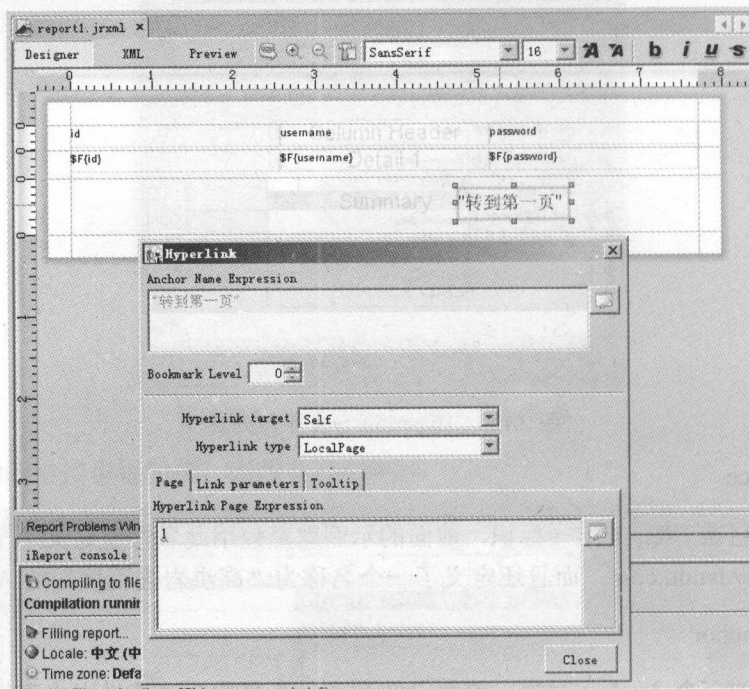


图 2.122 返回第 1 页

(4) RemoteAnchor

可以设置在导出的 HTML 文件中有一个链接，单击链接后到达 C 盘中的某一个 HTML，并且定义到指定的 Anchor，导出 HTML 定位 findme 的报表模板如图 2.123 所示。

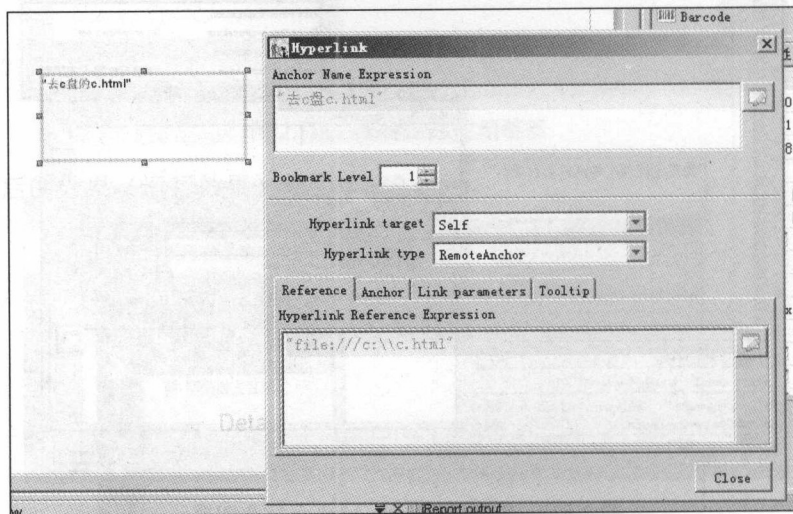


图 2.123 导出 HTML 定位 findme 的报表模板

继续设置目标 Anchor 的名称为 endend，如图 2.124 所示。

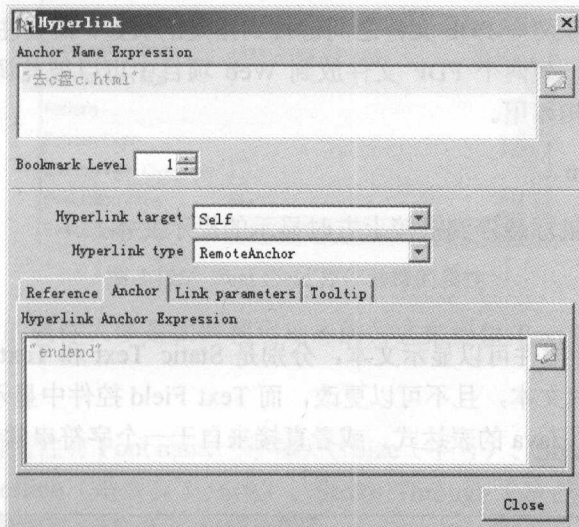


图 2.124 设置 Anchor 为 endend 值

文件 c.html 中有如下 Anchor 代码：

```
<a name="endend">asdf</a>
```

在 IE 中运行程序，单击链接转到 C 盘的 c.html，然后定位到名称为 endend 的 Anchor 上。需要说明的是，不是所有的文件格式都支持超链接的形式。

还可以用 HTTP 协议来访问远程 HTML 中的 Anchor，设计界面（使用 HTTP 协议访问远程 HTML 中的 Anchor）如图 2.125 所示。

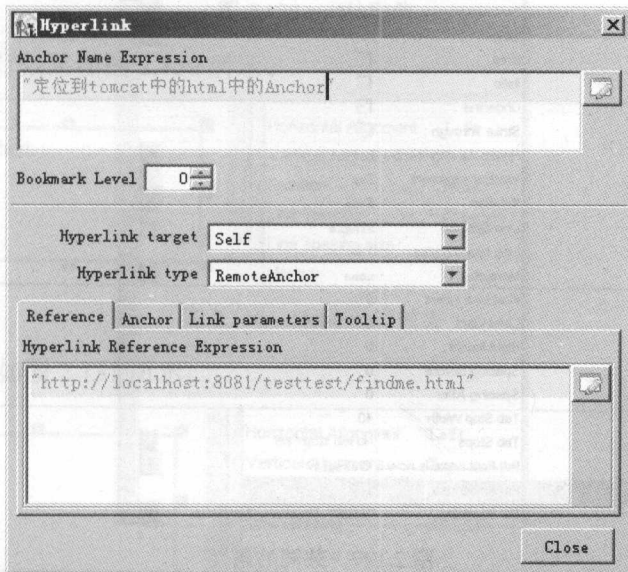


图 2.125 使用 HTTP 协议访问远程 HTML 中的 Anchor

(5) RemotePage

此属性值主要用于在两个 PDF 文件之间链接到指定的页数，需要注意的是这两个 PDF 文件必须放在同一个位置，如两个 PDF 文件放到 Web 项目中可以彼此调用，或者这两个 PDF 文件放到硬盘中可以互相调用。

(6) ReportExecution

该选项用于设置当鼠标悬停到链接上方时显示的提示文本。

2.4.4 文本控件

在 iReport 中有两种控件可以显示文本，分别是 Static Text 和 Text Field，它们的区别是 Static Text 只显示静态的文本，且不可以更改，而 Text Field 控件中显示的文本可以来自于变量、数据表字段中的值、Java 的表达式，或者直接来自于一个字符串常量，如图 2.126 所示为使用常量字符串。

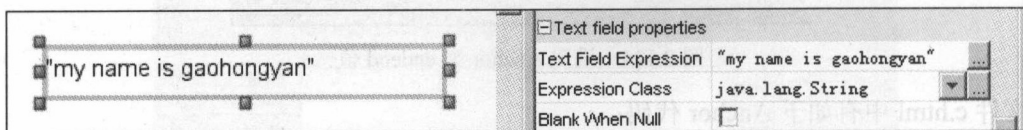


图 2.126 使用常量字符串

Text Field 控件支持 alignment（对齐）、position（位置）、line breaks（行回车）等功能，还支持随着内容的增加而在垂直方向上自动扩展的效果，以及支持超链接的效果。

控件 Static Text 和 Text Field 的共同属性如图 2.127 所示。

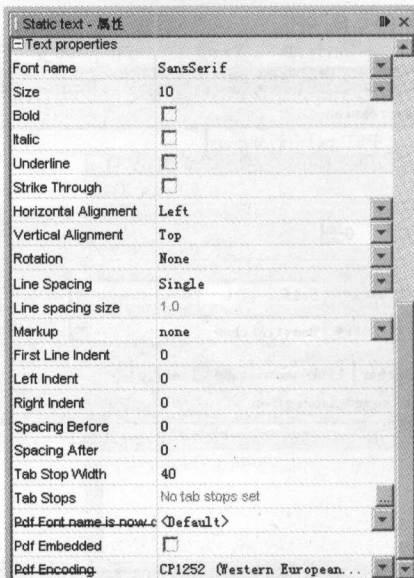


图 2.127 共同的属性

控件 Text Field 还具有自己独特的属性，如图 2.128 所示。

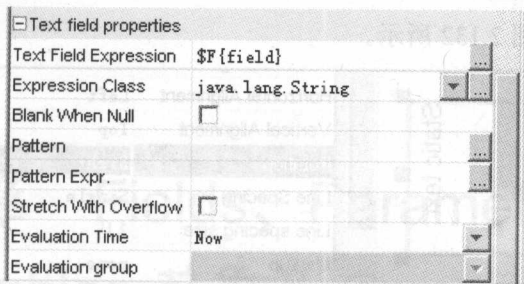


图 2.128 Text Field 控件独特的属性

在图 2.128 中显示的属性大多是如何利用动态的方式来提供 Text Field 显示的值。

1. 常用属性

这两个控件的常用属性有 Font name (字体)、Size (字号)、Bold (是否为粗体)、Italic (是否为斜体)、Underline (是否有下划线)、Strike Through (是否有删除线)、Horizontal Alignment (水平对齐方式)、Vertical Alignment (垂直对齐方式)。

2. Rotation 属性

属性 Rotation 的功能是定义文字如何打印，4 种取值如图 2.129 所示。

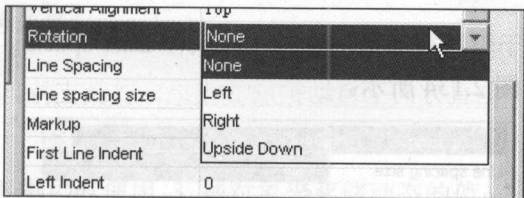


图 2.129 4 种取值

其中 None 的运行效果如图 2.130 所示。

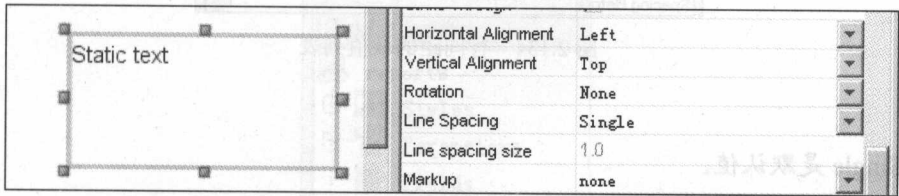


图 2.130 None 运行效果

Left 的运行效果如图 2.131 所示。

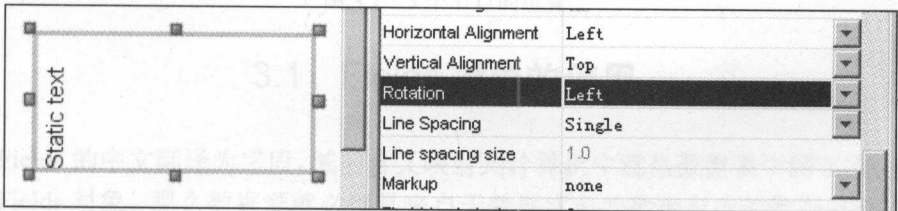


图 2.131 Left 运行效果

Right 的运行效果如图 2.132 所示。

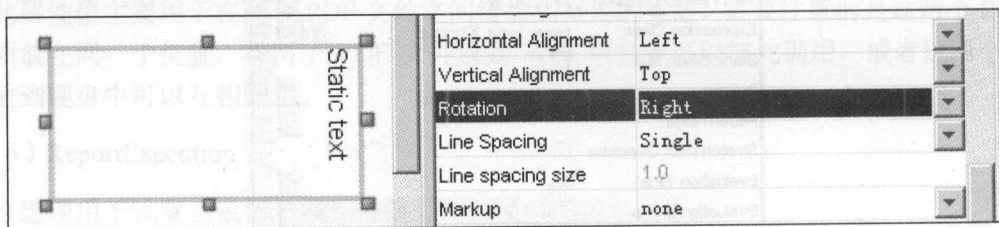


图 2.132 Right 运行效果

Upside Down 的运行效果如图 2.133 所示。

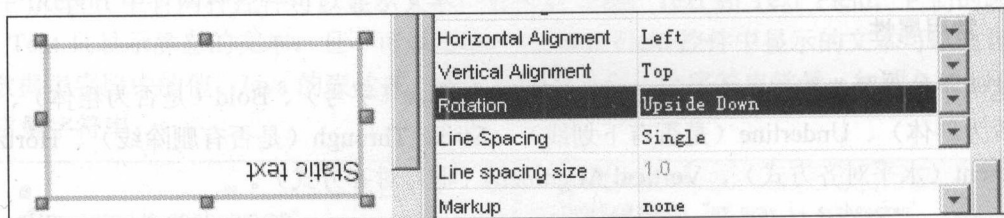


图 2.133 Upside Down 运行效果

3. Line Spacing 和 Line spacing size 属性

属性 Line Spacing 用于设置行间距的种类，而属性 Line spacing size 用于精确行间距的取值，行间距的取值列表如图 2.134 所示。

Line Spacing	Double
Line spacing size	Single
Markup	1.5
First Line Indent	Double
Left Indent	At least
Right Indent	Fixed
Spacing Before	Proportional

图 2.134 行间距的取值列表

其中：

- Single 是默认值。
- 1.5 表示 1.5 倍的间距。
- Double 表示双倍的间距。

第 3 章 Fields、Parameters、Variables 对象及 Group 分组

↓ 导言

本章将为大家介绍 JasperReports 框架中常用的 3 个对象，即 Fields、Parameters 及 Variables，以及 Group 分组。这 3 个对象在开发报表的过程中是非常重要的知识点，读者应该着重掌握如下内容：

- ★ Fields 对象的使用
- ★ Parameters 对象的使用
- ★ Variables 对象的使用
- ★ Group 分组的使用

在 JasperReports 框架中可以使用 3 种对象来存储动态的值：Fields、Parameters 以及 Variables 对象。

可以在 iReport 中添加这 3 个对象，如图 3.1 所示。

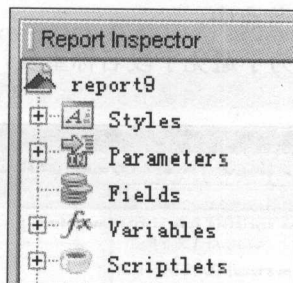


图 3.1 3 个节点的位置

3.1 Fields 对象的使用

对象 Fields 的中文翻译为字段，域的含义映射到计算机中就是数据表中的字段，也就是说，如果使用 Fields 对象，那么数据源就必须是来自于数据库中的数据表中的数据记录，当然，也有例外的情况，即它可以对应到某一个实体类中的某一个属性。

每一条记录都有不同的列,类似的情况是每一个实体都有不同的属性,若想显示出不同列、属性的值,那么就需要使用 Fields 对象了。

3.1.1 使用 Text Field 控件显示数据表字段值

其实 Fields 对象的使用在前面的章节已经有所涉及,利用 Report query 对话框就可以自动取出 Fields 对象,事实上,利用 Report query 对话框可以非常方便地取出一个表中的所有字段,如图 3.2 所示。

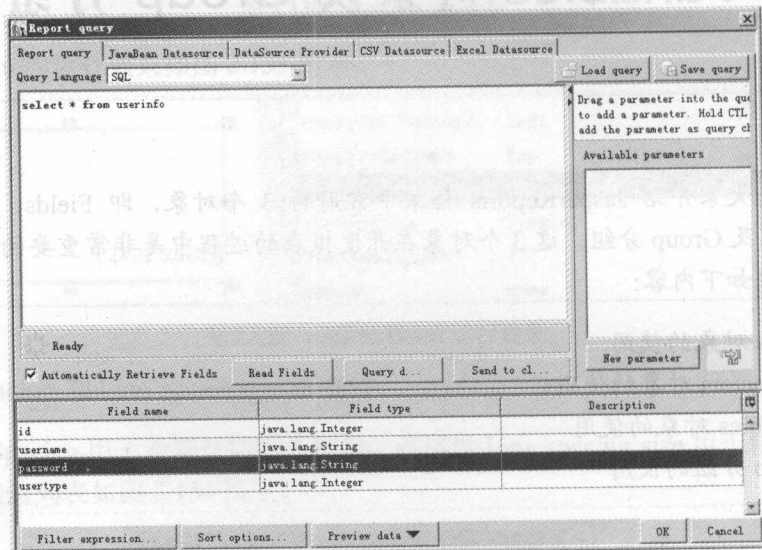


图 3.2 Report query 对话框

在 Report query 选项卡中输入 SQL 语句“select * from userinfo”后,将自动在下方显示出当前 SQL 语句查询出来的所有的 Field 字段名称及数据类型,单击 OK 按钮后就可以将当前的数据列添加到报表模板中的 Fields 节点中。

当然还可以使用多表查询,但为了避免字段名称重复,建议采用加入别名的方式,效果如图 3.3 所示。

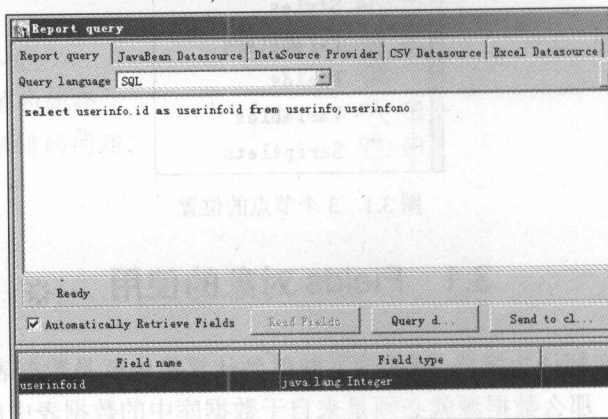


图 3.3 添加别名

获得 Fields 对象的下一步就是取出其中的值并打印出来,可以使用如下表达式来取得字段中的值:

```
$F{<field name>}
```

例如有 4 个 Fields, 手动拖曳一个 Text Field 控件到报表模板中, 设置其 Text Field Expression 属性值为 `$F{username}`, 也就是使用 Text Field 控件显示 username 字段的值, 如图 3.4 所示。

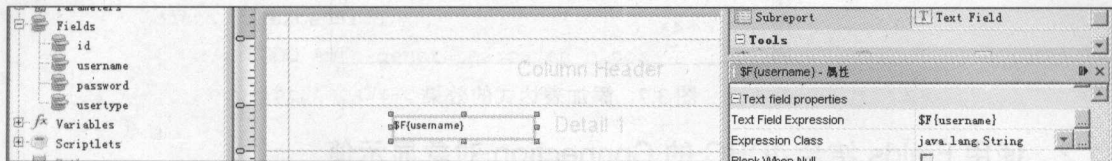


图 3.4 使用 Text Field 控件显示 Fields 对象的值

运行效果如图 3.5 所示。

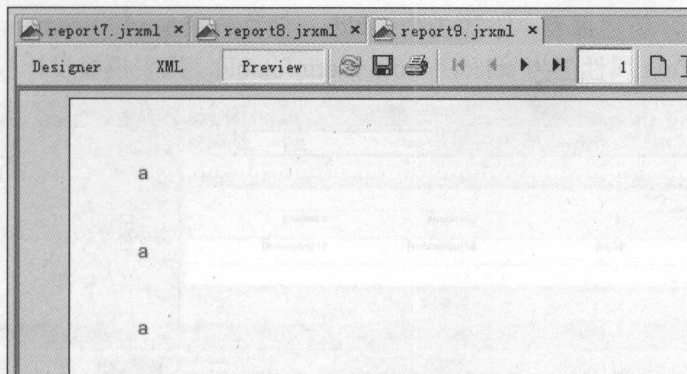


图 3.5 只显示 username 字段值

当然, 还可以在 Text Field Expression 属性中添加 Java 的表达式:

```
($F{username}.equals("a"))?"它是a":"它不是a"
```

属性设置如图 3.6 所示。

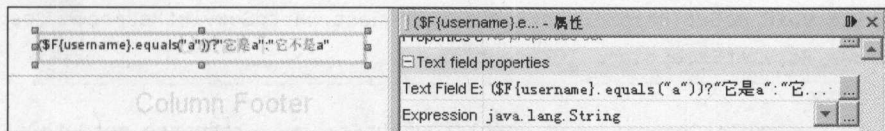


图 3.6 设置 Text Field Expression 属性 (含三元运算符)

运行效果如图 3.7 所示。

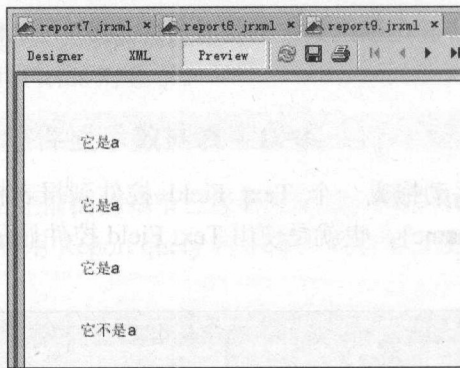


图 3.7 添加表达式的效果

3.1.2 使用 Fields 结合 JDBC 的 Connection 对象显示值

前面的章节是在 iReport 中使用 JDBC 打印 userinfo 表中的数据, 本示例将演示如何在 Web 项目中使用 JDBC 的 Connection 对象来打印 userinfotable 表中的数据。

创建 Web 项目, 新建报表, 添加 3 个 Fields, 名称分别是 id、username 和 password, 并设置报表的 Query Text 属性值为 select * from userinfotable, 报表模板设计界面如图 3.8 所示。

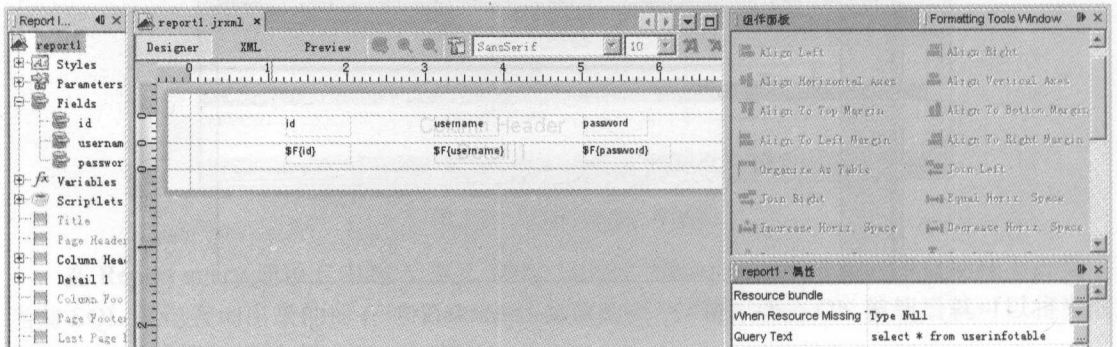


图 3.8 报表模板设计界面

Servlet 的核心代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report1.jasper");
            String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
            String username = "sa";
            String password = "";
            String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
```

```

        Class.forName(driverName);
        Connection connection = DriverManager.getConnection(url,username,password);
        JasperRunManager.runReportToPdfStream(reportStream,
        servletOutputStream, new HashMap(), connection);
        response.setContentType("application/pdf");
        servletOutputStream.flush();
        servletOutputStream.close();
    }
    catch (JRException e)
    { // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (ClassNotFoundException e)
    { // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (SQLException e)
    { // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

运行效果如图 3.9 所示。

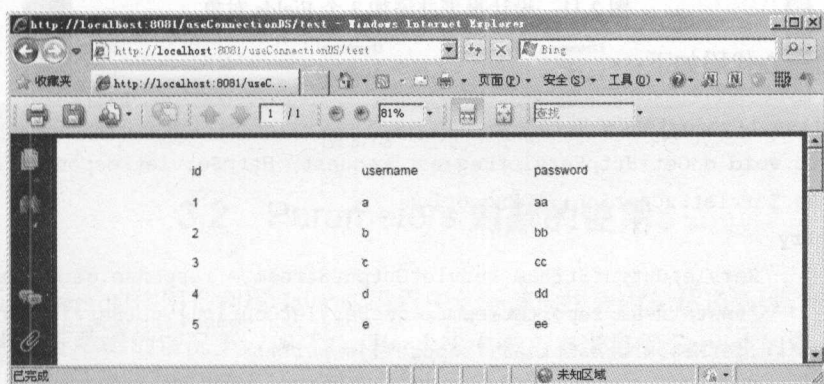


图 3.9 运行效果

3.1.3 使用 Fields 对象显示 Java 集合中实体类的属性值

其实 Fields 对象不仅支持从数据表中的指定字段打印值,还支持从 Java 的 Collection 对象的实体类中打印它的属性值。

创建 Web 项目,新建 Userinfo.java 实体类,该类具有三个属性及一个有参构造方法,类结构如图 3.10 所示。

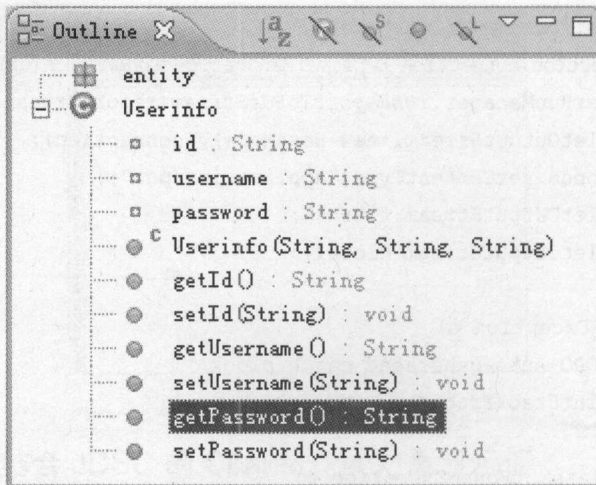


图 3.10 Userinfo.java 类的结构

创建报表并且添加三个 Fields 对象，即 id、username 以及 password 字段，如图 3.11 所示。

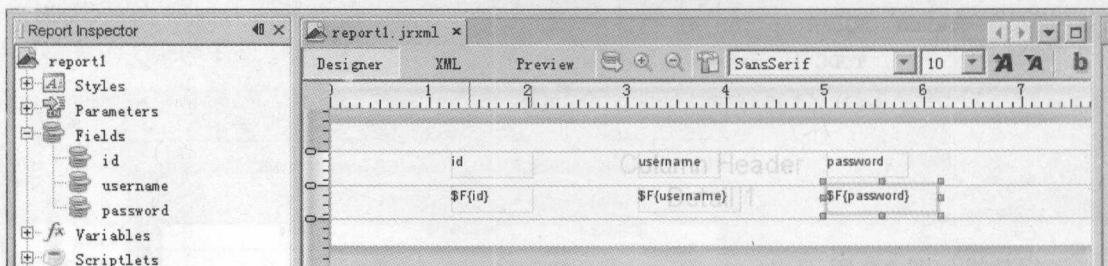


图 3.11 设计报表并添加 3 个 Fields 对象

Servlet 的核心代码如下：

```

public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext().
                getResourceAsStream("report1.jasper");
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 50; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    +(i+1),"password" + (i + 1)));
            }
            JasperRunManager.runReportToPdfStream(reportStream,
                servletOutputStream, new HashMap(),
                new JRBeanCollectionDataSource(listUserinfo));
            response.setContentType("application/pdf");
        }
    }
}

```



```

        servletOutputStream.flush();
        servletOutputStream.close();
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

运行效果如图 3.12 所示。

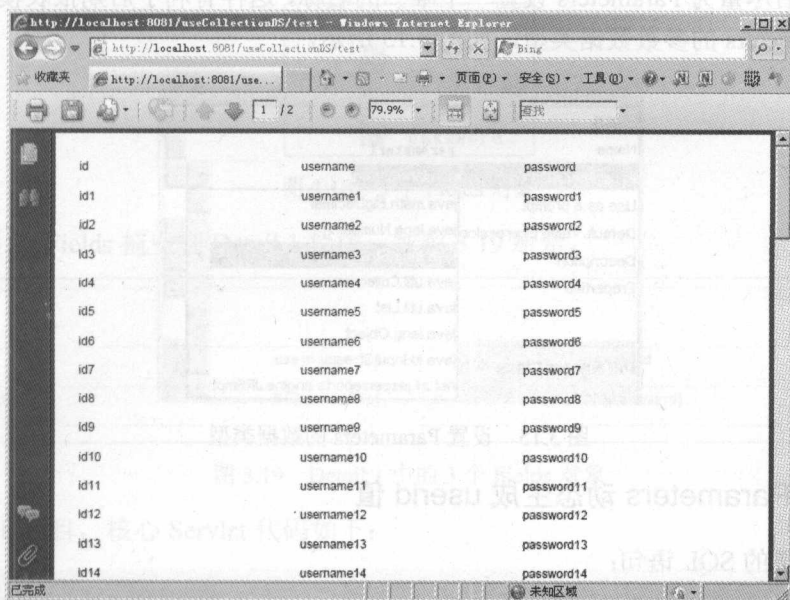


图 3.12 运行效果

3.2 Parameters 对象的使用

对象 Parameters 的作用是利用 Java 应用程序从外部向报表内部传递数据，此技术可以应用在任何动态处理数据的情况下，如 Title Band 栏中的文本来自于 Servlet 的定义，还可以动态创建 SQL 语句。

可以在 iReport 中定义 Parameters 对象，如图 3.13 所示。

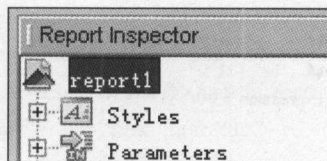


图 3.13 Parameters 结点

添加一个 Parameters 对象很简单, 只须单击鼠标右键, 在弹出的快捷菜单中选择“添加 Parameter”命令即可, 效果如图 3.14 所示。

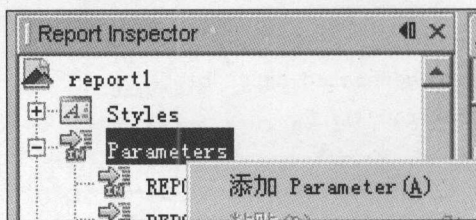


图 3.14 添加参数

添加完成后尽量为 Parameters 设置一个唯一的名称, 这样有利于后期报表模板的维护, 也可以设置 Parameters 的参数数据类型, 如图 3.15 所示。

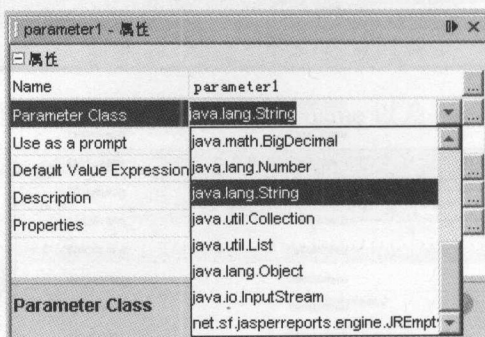


图 3.15 设置 Parameters 的数据类型

3.2.1 使用 Parameters 动态生成 userid 值

例如有如下的 SQL 语句:

```
select * from userinfo where id=2
```

此时想把“id=2”的值设计成一个动态传递的效果, 这时就要使用到 Parameters 对象, 那么它的 SQL 语句就要设计成如下的样式:

```
select * from userinfo where id=$P{userid}
```

新建一个报表模板, 添加一个名称为 userid 的 Parameters 参数对象, 如图 3.16 所示。

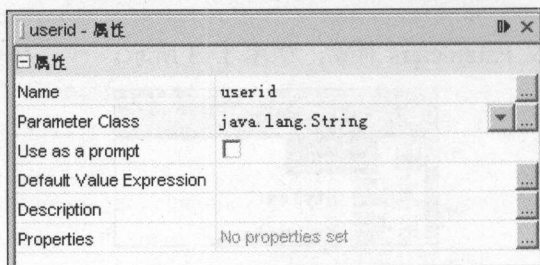


图 3.16 userid 属性设置

再设计报表的 SQL 语句, 如图 3.17 所示。

Query Text select * from userinfo table where id=\${F{userid}}

图 3.17 SQL 语句

生成的 XML 代码如下:

```
<queryString language="SQL">
  <![CDATA[select * from userinfo table where id=${F{userid}}]]>
</queryString>
```

添加 3 个 Fields 对象, 效果如图 3.18 所示。

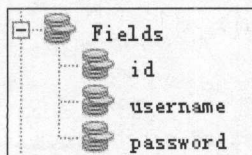


图 3.18 3 个 Fields 对象

并把这 3 个 Fields 拖曳到 Detail 1 栏中, 如图 3.19 所示。

Page Header		
Column Header		
id	username	password
Detail 1		
\${F{id}}	\${F{username}}	\${F{password}}

图 3.19 Detail 1 中的 3 个 Fields 对象

创建 Web 项目, 核心 Servlet 代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report2.jasper");
            String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
            String username = "sa";
            String password = "";
            String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
            Class.forName(driverName);
            Connection connection=DriverManager.getConnection(url, username,password);
            HashMap parameters = new HashMap();
            parameters.put("userid", 3);
            JasperRunManager.runReportToPdfStream(reportStream,
```



```

        servletOutputStream, parameters, connection);
        response.setContentType("application/pdf");
        servletOutputStream.flush();
        servletOutputStream.close();
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (ClassNotFoundException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (SQLException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

运行效果如图 3.20 所示。

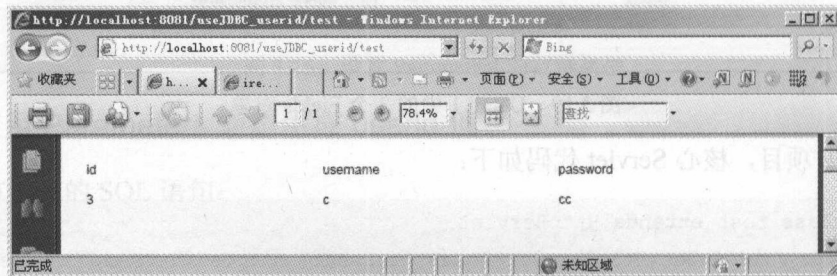


图 3.20 运行效果

3.2.2 使用 Parameters 动态生成 Date 区间的测试

使用 “\$P{参数名称}” 还可以查询 Date 区间。新建报表模板，添加两个参数，它们的参数类型都是 `java.sql.Timestamp`，Parameters 参数的属性设置如图 3.21 所示。

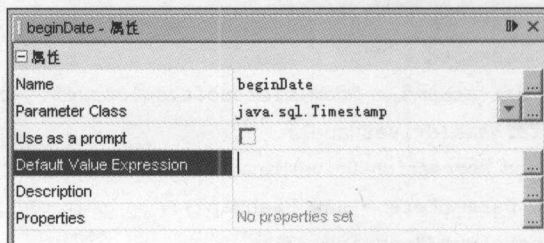


图 3.21 设置参数数据类型

设计报表使用的 SQL 语句，属性值定义如图 3.22 所示。

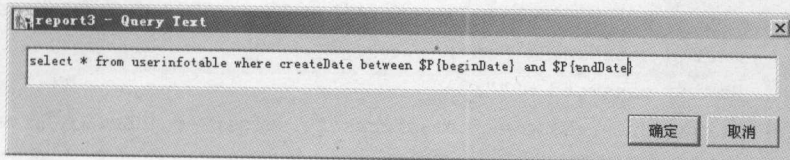


图 3.22 报表使用的 SQL 语句

数据表 userinfotable 的表结构如图 3.23 所示。

列名	数据类型	允许 Null 值
id	int	<input type="checkbox"/>
username	varchar(50)	<input checked="" type="checkbox"/>
password	varchar(50)	<input checked="" type="checkbox"/>
createDate	datetime	<input checked="" type="checkbox"/>
		<input type="checkbox"/>

图 3.23 userinfotable 表结构

数据表 userinfotable 中的数据内容如图 3.24 所示。

id	username	password	createDate
1	a	aa	2000-01-01 00:00:01.000
2	b	bb	2000-01-02 12:56:00.000
3	c	cc	2000-01-01 00:00:02.000
4	d	dd	2000-01-01 23:59:59.000
5	e	ee	2000-01-01 00:00:00.000
* NULL	NULL	NULL	NULL

图 3.24 数据表内容

在 iReport 中添加 4 个 Fields 对象，并将它们拖曳到 Detail 1 栏中，效果如图 3.25 所示。

id	username	password	createDate
\$F{id}	\$F{username}	\$F{password}	\$F{createDate}

图 3.25 Detail 1 中有 4 个 Fields 对象

新建 Web 项目，Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext().
```

```

        .getResourceAsStream("report3.jasper");
        String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
        String username = "sa";
        String password = "";
        String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
        Class.forName(driverName);
        Connection connection=DriverManager.getConnection(url, username,password);
        String beginDateString = "2000-1-1 00:00:00";
        String endDateString = "2000-1-1 23:59:59";
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
        Timestamp beginDate=new Timestamp(sdf.parse(beginDateString).getTime());
        Timestamp endDate = new Timestamp(sdf.parse(endDateString).getTime());
        HashMap parameters = new HashMap();
        parameters.put("beginDate", beginDate);
        parameters.put("endDate", endDate);
        JasperRunManager.runReportToPdfStream(reportStream,
        servletOutputStream, parameters, connection);
        response.setContentType("application/pdf");
        servletOutputStream.flush();
        servletOutputStream.close();
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (ClassNotFoundException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (SQLException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (ParseException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

项目运行效果如图 3.26 所示。

id	username	password	createDate
1	a	aa	2000-01-01 00:00:01.0
3	c	cc	2000-01-01 00:00:02.0
4	d	dd	2000-01-01 23:59:59.0
5	e	ee	2000-01-01 00:00:00.0

图 3.26 正确取出时间区间内的数据

3.2.3 使用 Parameters 动态生成 where 语句

使用 Parameters 对象还可以动态生成 where 语句。

创建一个报表，设计报表使用的 SQL 语句，如图 3.27 所示。

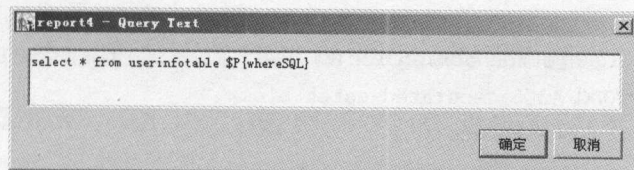


图 3.27 设置报表使用的 SQL 语句

在报表中继续添加 4 个 Fields 对象并添加到 Detail 1 栏中。

添加一个 Parameters 对象，Parameters 属性设置如图 3.28 所示。

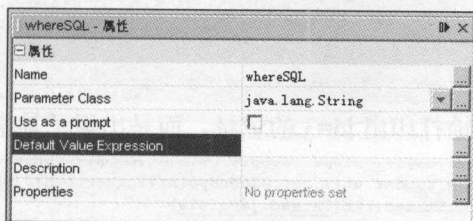


图 3.28 设置 whereSQL 属性参数

创建 Web 项目，Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report4.jasper");
            String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
            String username = "sa";
            String password = "";
```

```

String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
Class.forName(driverName);
Connection connection=DriverManager.getConnection(url, username,password);
HashMap parameters = new HashMap();
parameters.put("whereSQL", "where id=3");
JasperRunManager.runReportToPdfStream(reportStream,
servletOutputStream, parameters, connection);
response.setContentType("application/pdf");
servletOutputStream.flush();
servletOutputStream.close();
}
catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (ClassNotFoundException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (SQLException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

程序运行后并没有如期地打印出 id=3 的记录，而是出现了异常，效果如图 3.29 所示。

```

at org.apache.tomcat.util.net.JIoEndpoint$Worker.run(JIoEndpoint.java:445)
at java.lang.Thread.run(Thread.java:619)
Caused by: com.microsoft.sqlserver.jdbc.SQLServerException: '0P0' 附近有语法错误。
at com.microsoft.sqlserver.jdbc.SQLServerException.makeFromDatabaseError(U

```

图 3.29 出现了异常

为什么会出现这种情况呢？这是因为 iReport 不支持使用 Parameters 传递 where 或 group by 子句，如何解决呢？很简单，使用 \$P!{ParameterName} 即可，这种写法其实也就是把 SQL 语句的某个片段作为参数传入。

重新设计报表使用的 SQL 语句，如图 3.30 所示。

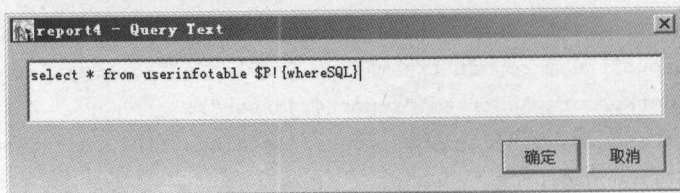
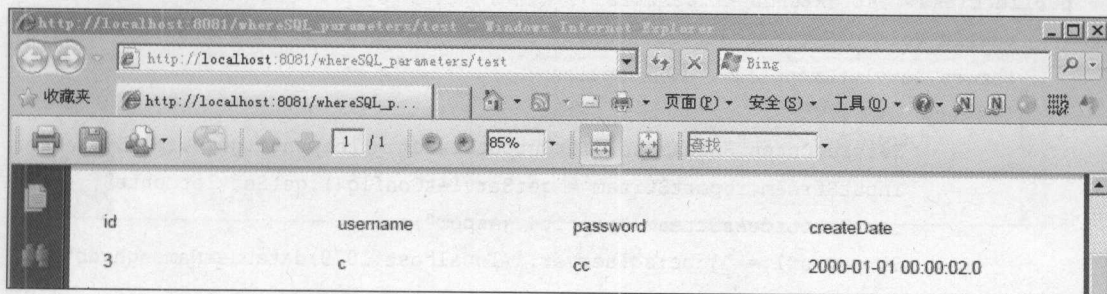


图 3.30 报表使用的最新 SQL 语句

从图 3.30 中可以看到,加了一个“!”感叹号,再次预览报表生成的.jasper 文件,放入 Web 项目的 WebRoot 文件夹中,重新运行 IE,正确显示出的结果如图 3.31 所示。



id	username	password	createDate
3	c	cc	2000-01-01 00:00:02.0

图 3.31 正确显示 id=3 的记录

因此,在动态创建 where 或 group by 这类 SQL 子句时,请使用 `$P!{ParametersName}` 格式的 SQL 语句。

3.2.4 使用 Parameters 对象实现 SQL 的 IN 及 NOTIN 查询

JasperReport 框架还支持 SQL 语句中的 IN 或 NOTIN 查询,使用如下语法格式:

```
$X{IN/NOTIN, FieldName, ParametersName}
```

其中 IN 或 NOTIN 是生成的 SQL 语句,而 FieldName 是对哪个字段进行 IN 操作,ParametersName 是 IN 语句使用的查询条件值,此值来自于 Parameters 对象。

新建报表,添加 4 个 Fields,并拖曳到 Detail 1 栏中。

设计报表使用的 SQL 语句,如图 3.32 所示。

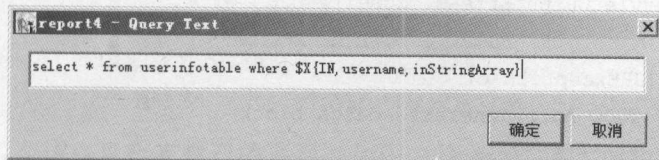


图 3.32 报表使用的 SQL 语句

注意报表使用的 SQL 语句为大写的 IN。

从图 3.32 中可以看到,操作是 IN,查询的字段是 username,而且要创建一个名称为 inStringArray 的 Parameters 对象,该对象的属性设置如图 3.33 所示。

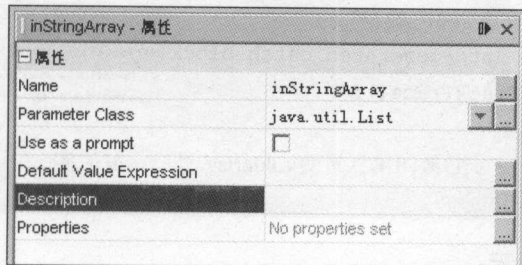
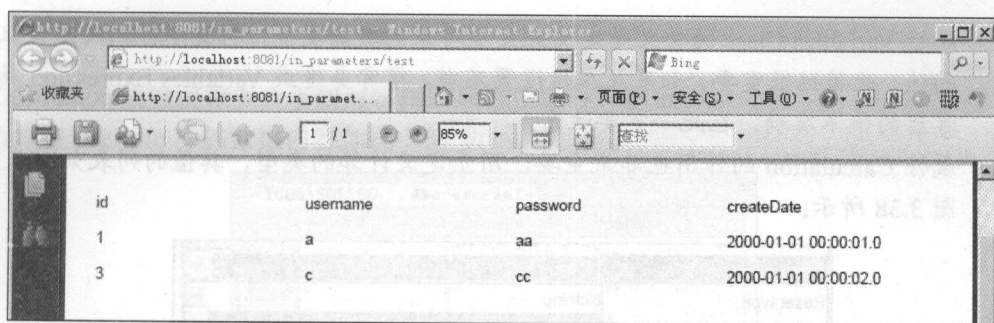


图 3.33 inStringArray 属性设置

新建 Web 项目，Servlet 核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            ServletOutputStream servletOutputStream = response.getOutputStream();
            InputStream reportStream = getServletConfig().getServletContext()
                .getResourceAsStream("report4.jasper");
            String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
            String username = "sa";
            String password = "";
            String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
            Class.forName(driverName);
            Connection connection=DriverManager.getConnection(url, username,password);
            List inList = new ArrayList();
            inList.add("a");
            inList.add("c");
            HashMap parameters = new HashMap();
            parameters.put("inStringArray", inList);
            JasperRunManager.runReportToPdfStream(reportStream,
                servletOutputStream, parameters, connection);
            response.setContentType("application/pdf");
            servletOutputStream.flush();
            servletOutputStream.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (ClassNotFoundException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (SQLException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

运行结果如图 3.34 所示。



id	username	password	createDate
1	a	aa	2000-01-01 00:00:01.0
3	c	cc	2000-01-01 00:00:02.0

图 3.34 运行效果

3.3 Variables 对象的使用

通过前面的学习可知, Fields 用于取得数据表中字段的值以及实体类属性的值, Parameters 用于取得从 Servlet 传递过来的值, 而 Variables 对象的作用是把表达式计算后的值存放在其内部, 也就是和编程语言中的变量具有同样的效果, 它的使用格式是 \$V{<variable name>}。

默认情况下, 报表引擎从 DataSource 数据源中每取出一条记录就把 Variables1 变量的值更新一次。

创建 Variables 对象很简单, 只须单击鼠标右键, 在弹出的快捷菜单中选择“添加 Variable”命令即可, 如图 3.35 所示。

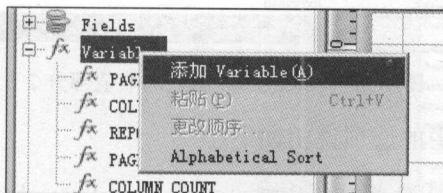


图 3.35 创建一个 Variables 变量对象

创建一个变量后, 该变量也有自己的属性, 如图 3.36 所示。

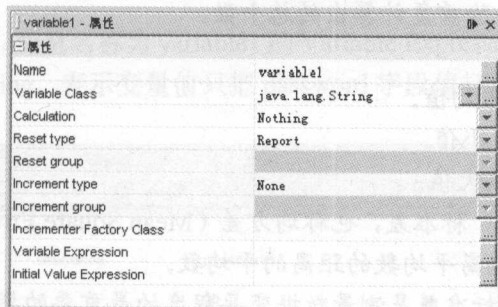


图 3.36 一个 Variables 变量对象的属性

其中:

- 属性 Name 的含义是定义此 Variables 变量对象的名称。

- 属性 Variables Class 的含义是定义此变量对象的数据类型，也就是存放的是什么类型的值，此属性很重要，如果此属性设置不当，很容易造成 Variables Expression 属性计算后的值不准确，在此一定要注意。
- 属性 Calculation 的作用也非常重要，用于定义计算的类型，其值的列表如图 3.37 和图 3.38 所示。

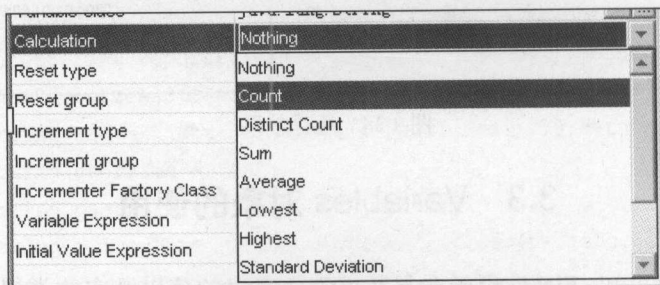


图 3.37 Calculation 属性的取值部分 1

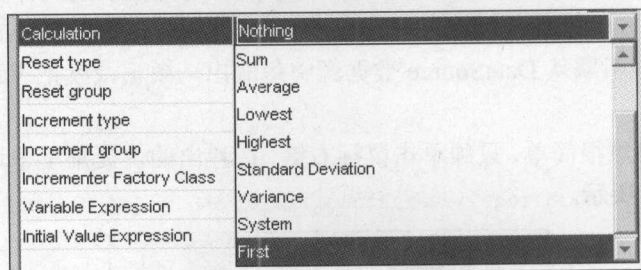


图 3.38 Calculation 属性的取值部分 2

对 Calculation 下拉列表中的选项说明如下。

- Nothing: 不做任何计算类型的处理，变量 Variables 的值来自于 Variable Expression 表达式的结果值。
- Count: 计算出总个数。
- Distinct Count: 去除重复计算出的总个数。
- Sum: 计算出汇总。
- Average: 计算出平均值。
- Lowest: 计算出最小值。
- Highest: 计算出最大值。
- Standard Deviation: 标准差，也称均方差 (Mean Square Error)，此值主要用在统计学中，它是各数据偏离平均数的距离的平均数。
- Variance: 方差和标准差是测量数据变异程度的最重要的指标，方差是各个数据与其算术平均数的差的平方和的平均数。
- System: 由 Java 代码进行控制，通常用于报表扩展。
- First: 来自于第 1 个值。

3.3.1 Calculation 属性

Calculation 属性值很多，下面对常用的属性值进行详细说明。
数据表 userinfo 中的数据内容如图 3.39 所示。

TC06\SQL200...dbo.userinfo			
	id	username	password
	1	a	aa
	2	b	bb
	3	c	cc
	4	a	aa
*	NULL	NULL	NULL

图 3.39 userinfo 数据表中的数据

1. Nothing

此属性值的含义是变量的值来自于 Variable Expression 表达式的结果值，设计报表并添加 Fields 及 Variables 对象如图 3.40 所示。

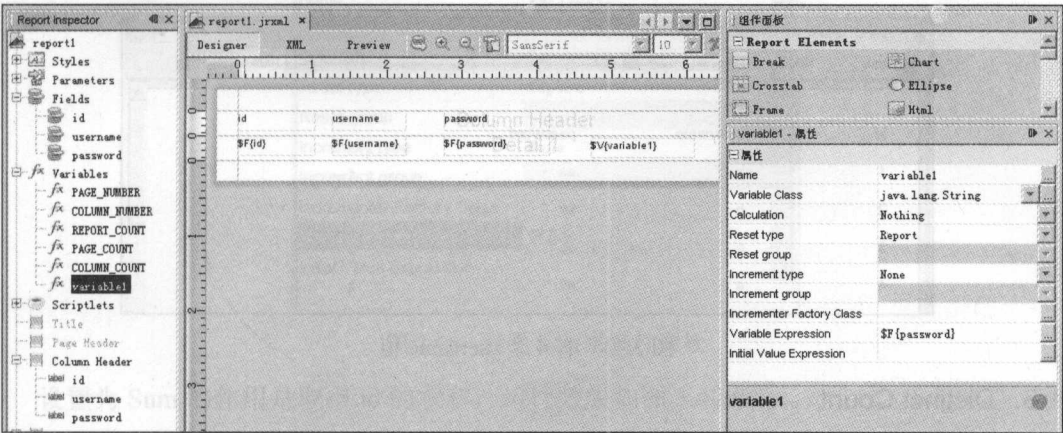


图 3.40 添加 Fields 和 Variables 对象的报表

从图 3.40 中可以看到，变量名称为 variable1 的 Variable Expression 属性值为 \$F{password}，Calculation 被设置为 Nothing，表示变量值只把 password 字段值打印出来，运行效果如图 3.41 所示。

The screenshot shows the 'Report Designer' window with the 'Preview' tab selected. The report displays a table with columns 'id', 'username', and 'password'. The 'password' column contains the values 'aa', 'bb', 'cc', and 'aa' for rows 1 through 4 respectively. The 'id' column contains values 1, 2, 3, and 4. The 'username' column contains values 'a', 'b', 'c', and 'a'.

图 3.41 只打印 password 字段值

2. Count

名称为 variable1 的 Variables 对象的属性设置如图 3.42 所示。

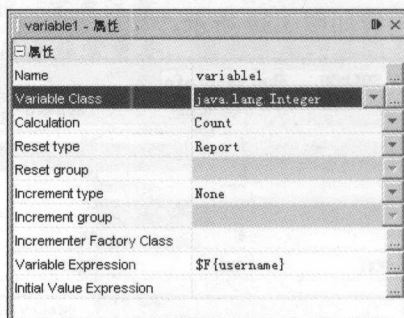


图 3.42 Count 的属性测试

从图 3.42 中可以看到, Variable Class 变量存值的数据类型为 java.lang.Integer, Calculation 设置为 Count, Variable Expression 设置为 \$F{username}, 也就是计算有多少个 username, 程序运行效果如图 3.43 所示。

The image shows the 'report1.jrxml' preview window. It displays a table with 4 rows of data. The columns are 'id', 'username', 'password', and a fourth column representing the count. The data is as follows:

id	username	password	
1	a	aa	1
2	b	bb	2
3	c	cc	3
4	a	aa	4

图 3.43 有 4 个 username 值

3. Distinct Count

重新设置属性值, 如图 3.44 所示。

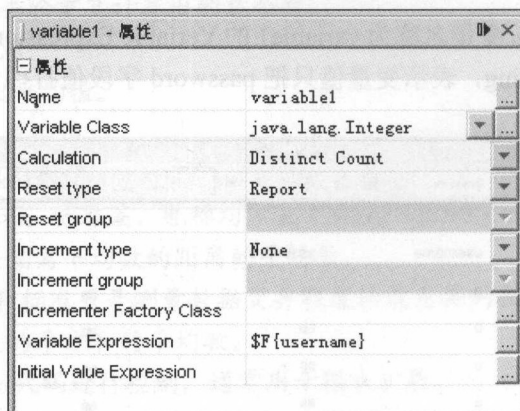
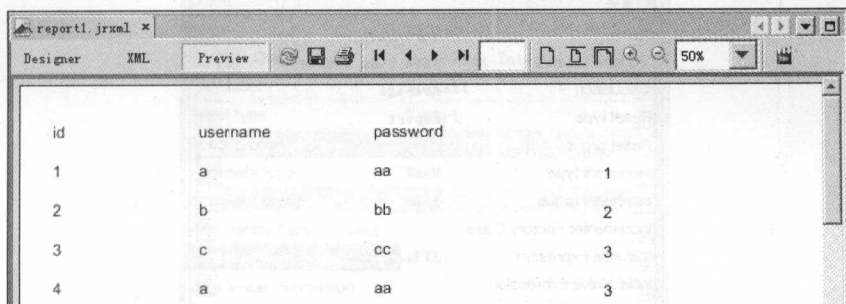


图 3.44 设置为 Distinct Count

设置为 Distinct Count 的作用是取得不重复的 username 值，运行效果如图 3.45 所示。

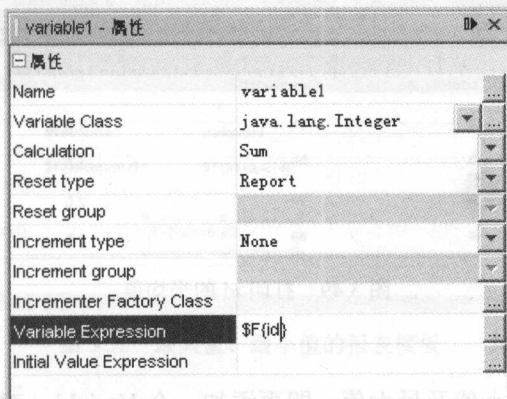


id	username	password	
1	a	aa	1
2	b	bb	2
3	c	cc	3
4	a	aa	3

图 3.45 不重复的 username 数量为 3

4. Sum

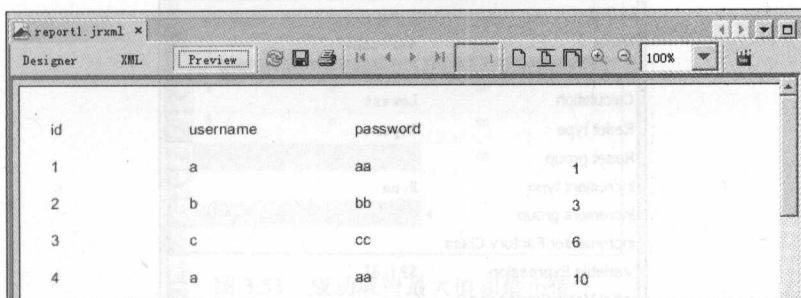
重新设置属性值，如图 3.46 所示。



variable1 - 属性	
Name	variable1
Variable Class	java.lang.Integer
Calculation	Sum
Reset type	Report
Reset group	
Increment type	None
Increment group	
Incrementer Factory Class	
Variable Expression	<code>\$F{id}</code>
Initial Value Expression	

图 3.46 设置 Sum 的属性面板

设置为 Sum 的作用是取得 id 的求和，运行效果如图 3.47 所示。



id	username	password	
1	a	aa	1
2	b	bb	3
3	c	cc	6
4	a	aa	10

图 3.47 运行效果

5. Average

设置为 Average 的含义是求平均值，属性面板设置如图 3.48 所示。

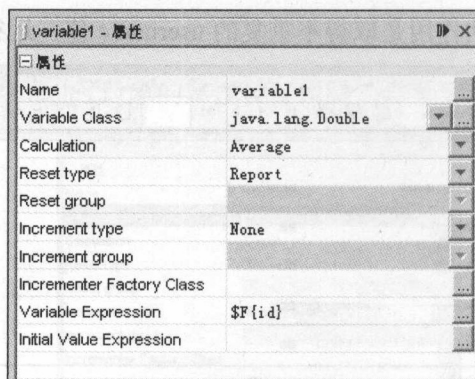


图 3.48 属性面板设置

运行效果如图 3.49 所示。

id	username	password	
1	a	aa	1.0
2	b	bb	1.5
3	c	cc	2.0
4	a	aa	2.5

图 3.49 打印 id 的平均值

6. Lowest 和 Highest

本示例将求出 id 的最小值及最大值，即再添加一个 Variables 对象，此时一共有两个变量对象，变量对象 variable1 的属性设置如图 3.50 所示。

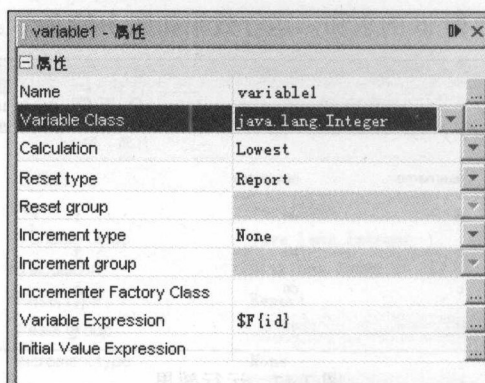


图 3.50 variable1 对象的属性

variable2 对象的属性设置如图 3.51 所示。

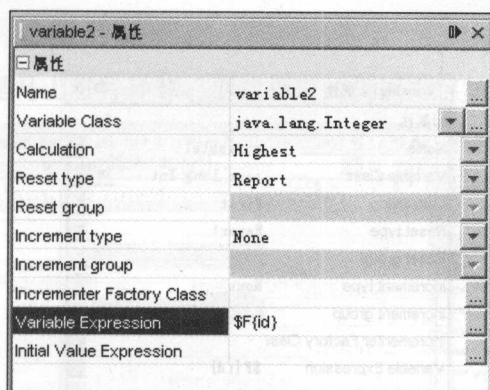


图 3.51 variable2 对象的属性

设计报表模板如图 3.52 所示。

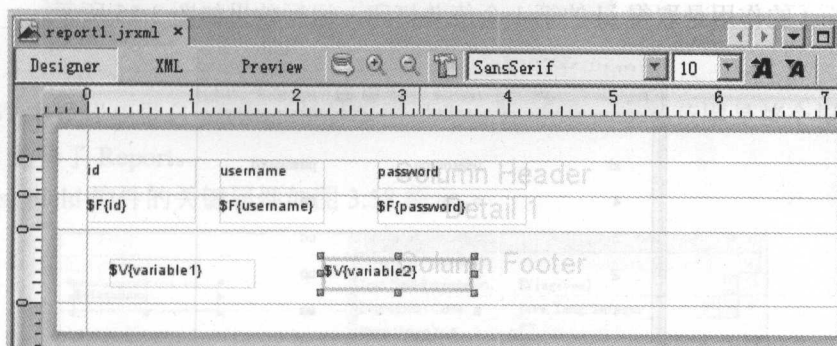


图 3.52 最大值、最小值的报表模板

运行效果如图 3.53 所示。

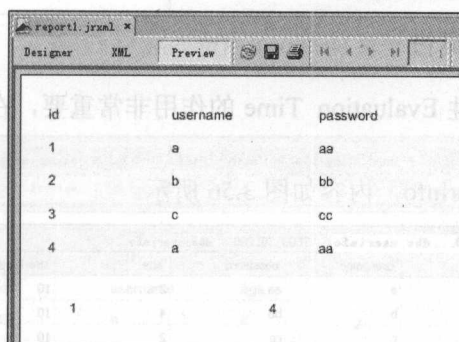


图 3.53 成功取得最大值和最小值

7. First

属性值 First 代表取得第 1 条记录中的值。

将报表的 SQL 语句改为如下形式：

```
select * from userinfo order by id desc
```

设置属性如图 3.54 所示。

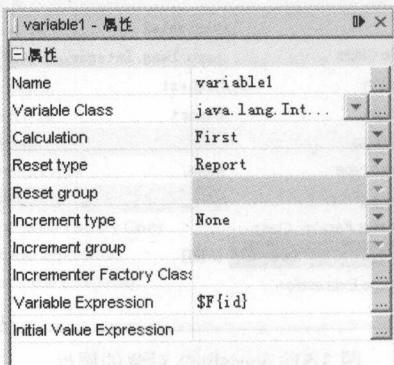


图 3.54 设置为 First

设置为 First 的作用是取得 id 的第 1 个值并打印，运行效果如图 3.55 所示。

id	username	password
4	a	aa
3	c	cc
2	b	bb
1	a	aa

图 3.55 打印第 1 个值——4

3.3.2 Evaluation Time 属性

首先需要强调一下，属性 Evaluation Time 的作用非常重要，在后面的章节中会有它使用的示例。

创建实验用的数据表 userinfo，内容如图 3.56 所示。

id	username	password	age	usertype
1	a	aa	2	10
2	b	bb	4	10
3	c	cc	2	10
4	d	dd	4	20
5	e	ee	5	20
6	f	ff	6	30

图 3.56 userinfo 数据表的内容

该属性的作用是设置什么时机对 Text Field Expression 属性中的表达式进行计算。

1. Now

默认值是 Now，也就是马上进行计算，创建报表的模板内容如图 3.57 所示。

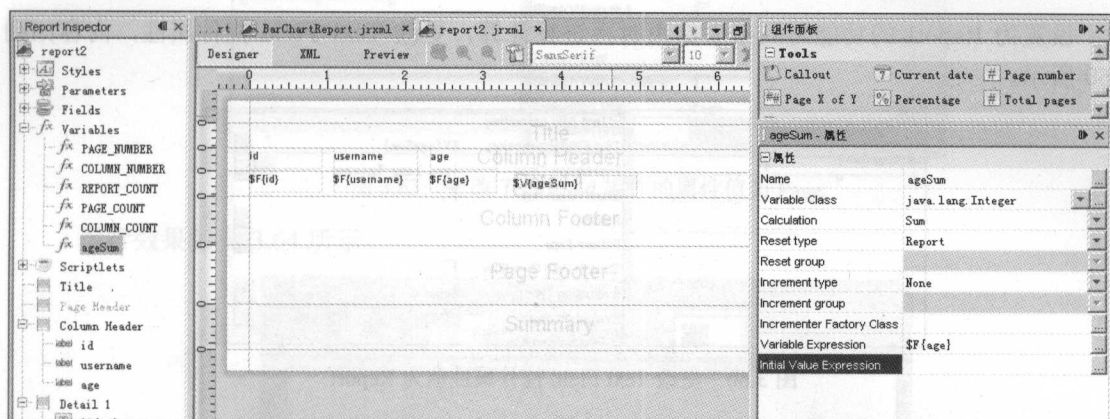


图 3.57 创建 Now 值报表模板

在图 3.57 中设置变量名 ageSum 的计算类型为 Sum 求和，求和范围是整个报表，因为属性 Reset type 选择了 Report。

其中 Text Field 控件的关键属性如图 3.58 所示。

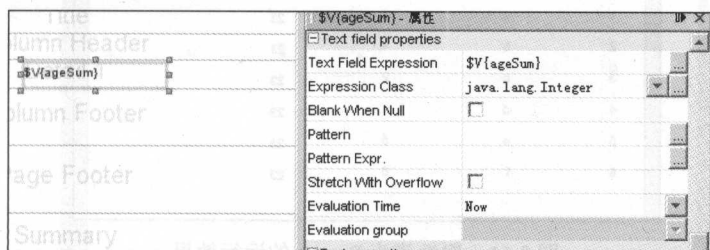


图 3.58 Now 值的 Text Field 属性

运行效果如图 3.59 所示。

id	username	age	ageSum
1	a	2	2
2	b	4	6
3	c	2	8
4	d	4	12
5	e	5	17
6	f	6	23

图 3.59 Now 值的运行效果

通过图 3.59 可以发现，每读一行数据时就把 age 的值进行了自动增加。

2. Report

设置为 Report 的功能是到报表结束时再进行计算，更改 Text Field 控件属性，内容如图 3.60 所示。

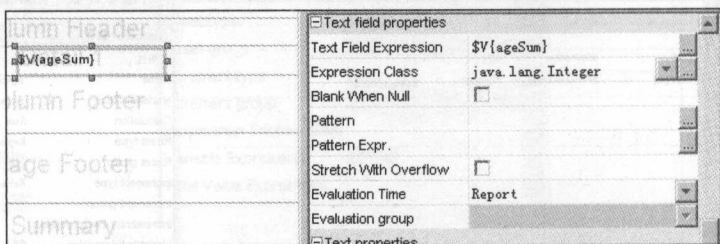


图 3.60 更改 Text Field 控件属性值为 Report

报表运行效果如图 3.61 所示。

id	username	age	
1	a	2	23
2	b	4	23
3	c	2	23
4	d	4	23
5	e	5	23
6	f	6	23

图 3.61 属性值为 Report 的运行效果

3. Page

设置为 Page 的功能是到本页 Page 的结束再进行计算，也就是把当前 Page 页中的数据都填充完毕再进行计算，更改报表的模板布局，如图 3.62 所示。

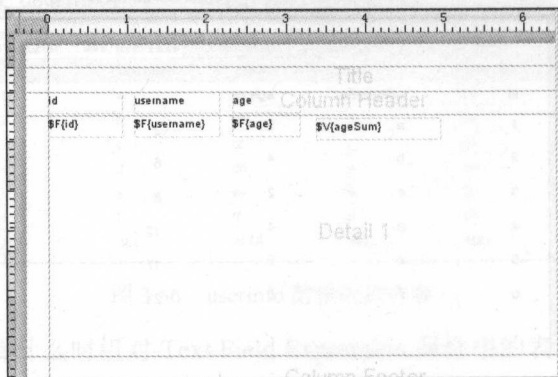


图 3.62 对 Detail 1 的 Band 高度进行增加

更改 Text Field 控件属性，内容如图 3.63 所示。

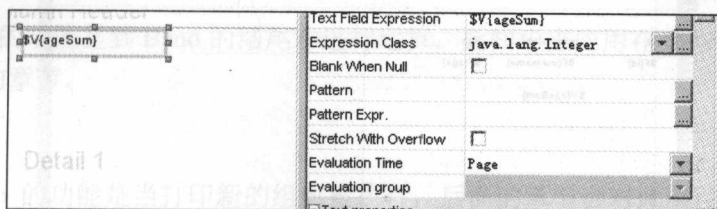


图 3.63 更改 Text Field 控件的属性值为 Page

运行效果如图 3.64 所示。

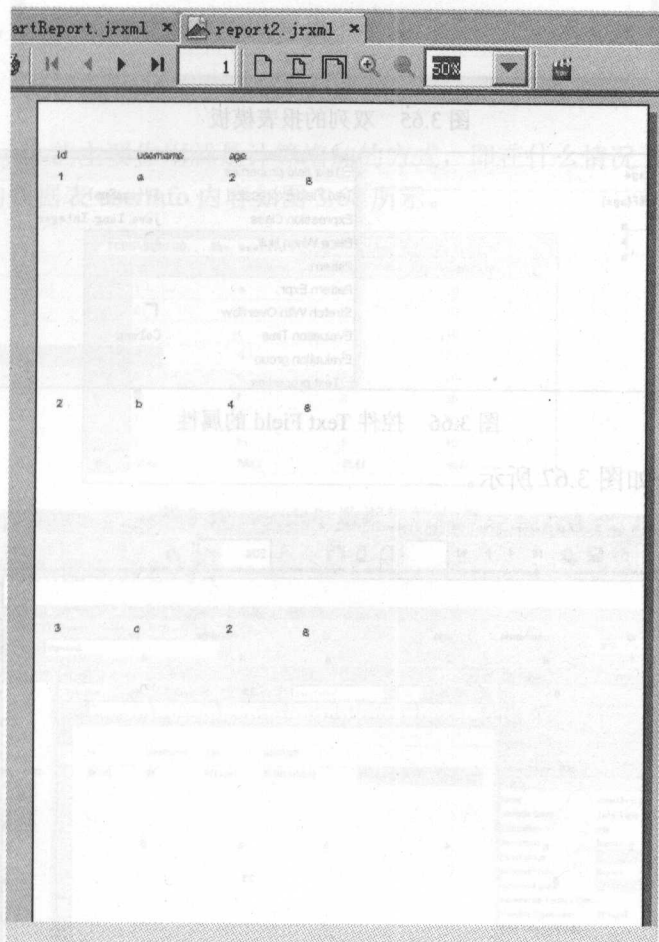


图 3.64 本页结束时计算 Sum 的值

4. Column

设置为 Column 的功能是新建一个 Column 列并填充里面的数据后再进行计算，更改报表模板布局，如图 3.65 所示。

控件 Text Field 的属性如图 3.66 所示。

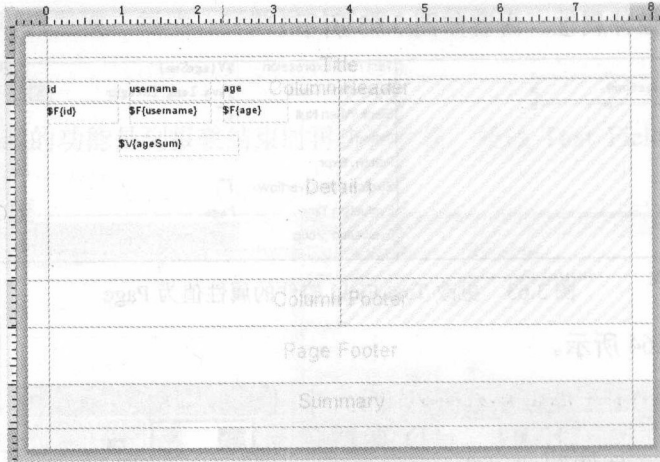


图 3.65 双列的报表模板

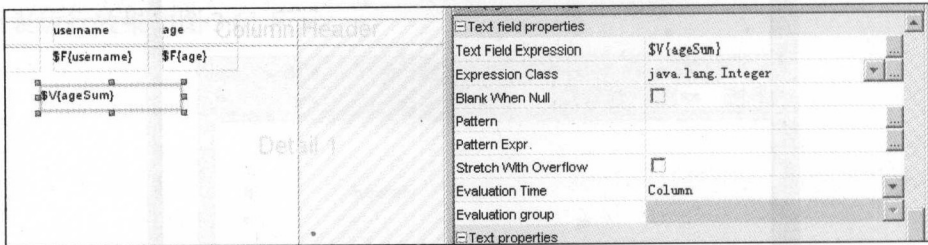


图 3.66 控件 Text Field 的属性

程序运行的效果如图 3.67 所示。

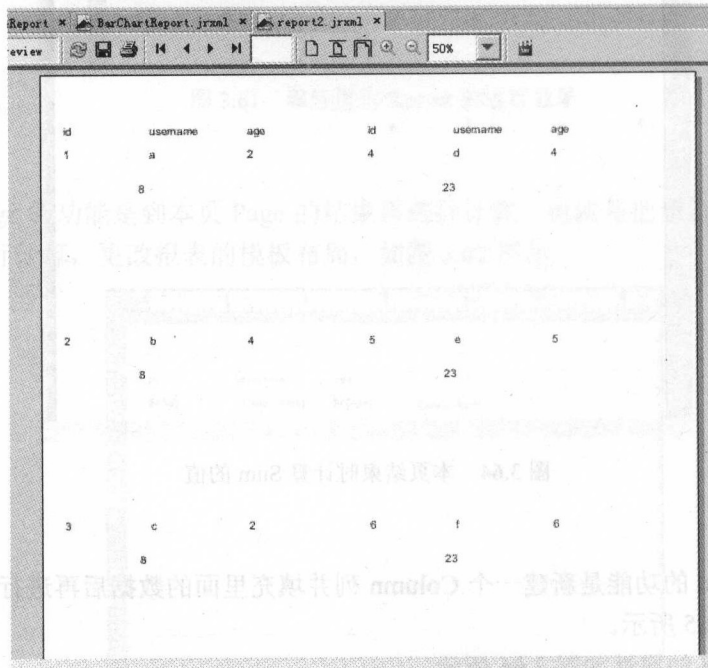


图 3.67 双列时每一列单独进行 Sum 运算

5. Band

设置为 Band 的功能是到 Band 的结尾才进行运算，此知识点应用在 Subreport 子报表中，具体请参看后面的章节。

6. Group

设置为 Group 的功能是当打印新的组时被执行，后面的章节会对此属性以示例的方式进行展示，这里不再赘述。

7. Auto

设置为 Auto 时，由于影响效率，所以建议不要使用。

3.3.3 Increment type 属性

属性 Increment type 的主要作用就是计算增加的方式，即在什么情况下进行增加。

本实验将使用的数据表 userinfo 内容如图 3.68 所示。

id	username	age	usertype
1	a	2	10
2	b	4	10
3	c	6	10
4	d	5	20
5	e	7	20
6	f	8	30
7	g	4	30
8	h	9	40
*	NULL	NULL	NULL

图 3.68 userinfo 数据表的内容

设计报表模板内容如图 3.69 所示。

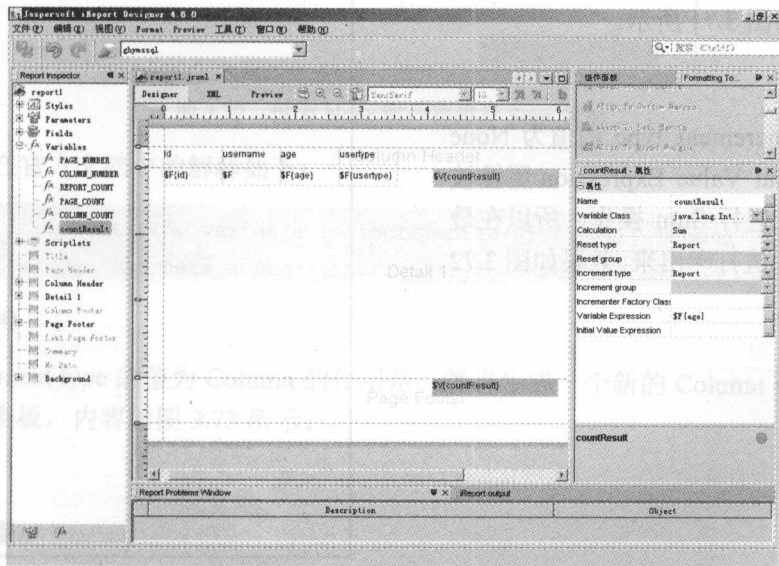


图 3.69 设计报表模板外观

1. Report

添加一个名称为 `countResult` 的 `Parameters` 变量对象，设计属性如图 3.70 所示。

在图 3.70 中设置的 `countResult` 变量属性用于统计 `Sum` 值，所以变量 `countResult` 的数据类型是 `java.lang.Integer`，`Result type` 为 `Report` 代表变量不会重置，是在整个处理报表的过程中一直 `Sum` 操作，`Sum` 操作什么呢？当然是 `Variable Expression` 属性的表达式 `$F{age}`，即年龄，也就是对 `age` 字段进行 `Sum` 操作。

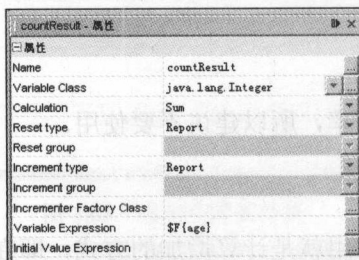


图 3.70 添加变量 `countResult` 属性设置

`Increment type` 的值为 `Report` 代表变量会在报表结束前打印最后一条记录的 `age` 值，也就是在报表处理的最后执行 `Sum` 操作，此属性值在官方文档中的解释如下：

Report-level increment: The variable never gets incremented during the report filling process (incrementType="Report").

运行效果如图 3.71 所示。

2. None

如果属性 `Increment type` 的值为 `None` 时，也就是 `Initial Value Expression` 属性被忽略，并且一直进行 `Sum` 操作，所以在最后把 `age` 的 `Sum` 值打印出来，效果如图 3.72 所示。

id	username	age	usertype
7	g	4	30
8	n	9	40
9			

图 3.71 值为 `Report` 的 `Sum` 值最后显示

id	username	age	usertype	
7	g	4	30	35
8	h	9	40	45

图 3.72 值为 None 时一直进行 Sum 累加

该属性值在官方文档中的解释如下：

Row-level increment: The variable is incremented with every record during the iteration through the data source (incrementType="None").

3. Column

属性 Increment type 的值为 Column 的作用是：每当生成一个新的 Column 列时进行增加，重新设计报表模板，内容如图 3.73 所示。

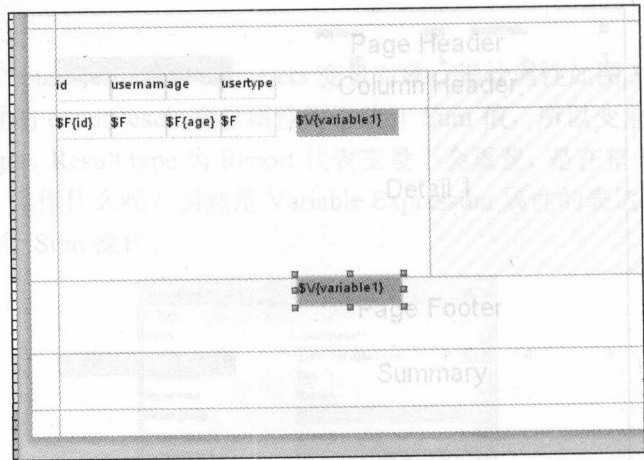


图 3.73 双列的模板

运行效果如图 3.74 所示。

The screenshot shows the rendered report with two columns of data. The data is organized into two separate tables side-by-side.

id	usernameage	usertype
1	a	2
2	b	4

id	usernameage	usertype
3	c	6
4	d	5

图 3.74 值为 Column 的第 1 页

在图 3.74 中可以看到, 当 id=2 时的 age=4 加上 id=4 时的 age=5, 结果为 9, 也就是在每创建一个新的 Column 时, 计算当前 row 的 age 值的 Sum。再来看第 2 页, 如图 3.75 所示。

id	username	age	usertype		id	username	age	usertype	
5	e	7	20	16	7	g	4	30	21
6	f	8	30	17	8	h	9	40	26
				17					26

图 3.75 值为 Column 的第 2 页

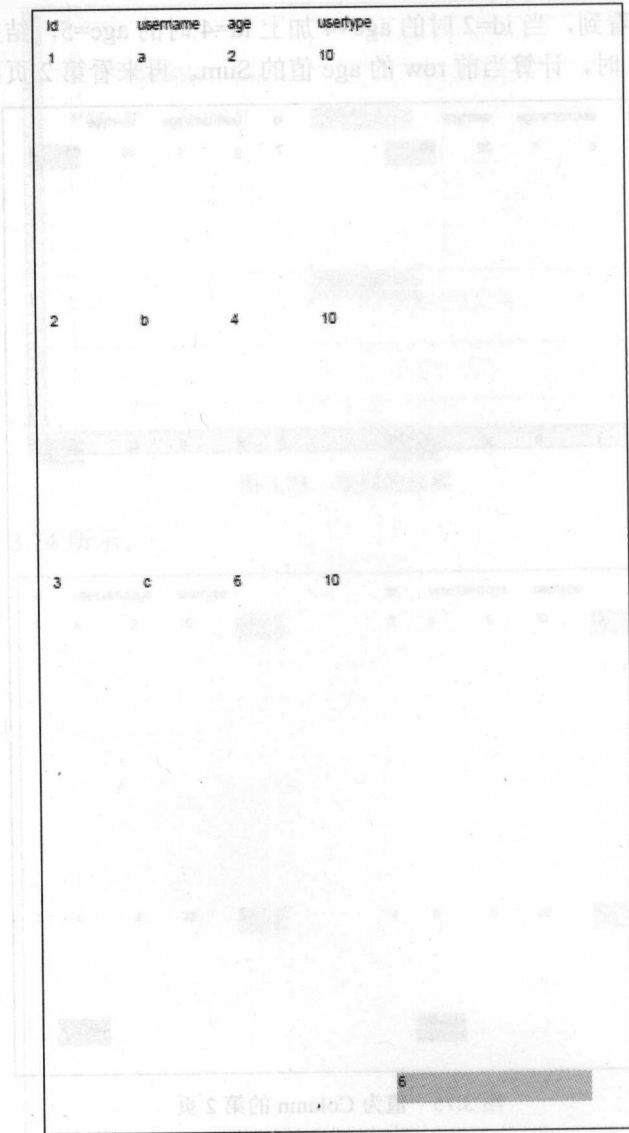
第 1 页执行完毕后 countResult 变量的值为 9, 9 加上 id=6 时的 age=8, 结果为 17, 依次类推, 这就是值为 Column 的运行效果。

该属性值在官方文档中的解释如下:

Column-level increment: The variable is incremented with each new column (incrementType="Column").

4. Page

值为 Page 和 Column 的运行效果大体相同, 只不过一个是 Column 级别, 另一个是 Page 级别, 运行效果如图 3.76 所示。



id	username	age	usertype
1	a	2	10
2	b	4	10
3	c	6	10

图 3.76 值为 Page 的第 1 页

继续查看第 2 页的运行效果，如图 3.77 所示。

id	username	age	usertype
4	d	5	20
5	e	7	20
6	f	8	30

图 3.77 值为 Page 的第 2 页

继续查看第 3 页的运行效果，如图 3.78 所示。

该属性值在官方文档中的解释如下：

Page-level increment: The variable is incremented with each new page (incrementType= "Page").

最后一个属性值 Group 将在后面的章节中为大家介绍，这里不再赘述。

id	username	age	usertype
7	g	4	30
8	h	9	40

23

图 3.78 值为 Page 的第 3 页

3.4 Group 分组的使用

在前面的章节中均在创建报表时使用 **Launch Report Wizard** 的方式来新建报表,在此向导中可以对报表进行分组,其实还可以先创建一个普通的报表,然后再设置分组。

3.4.1 Group 分组的使用方法

新建数据表 userinfo, 数据内容如图 3.79 所示。

	id	username	password	usertype
▶	1	a	aa	1
	2	b	bb	1
	3	c	cc	2
	4	d	dd	2
	5	e	ee	2
	6	f	ff	3
*	NULL	NULL	NULL	NULL

图 3.79 userinfo 数据表内容

usertype 表示要分组的字段，新建报表添加 4 个 Fields 对象，如图 3.80 所示。

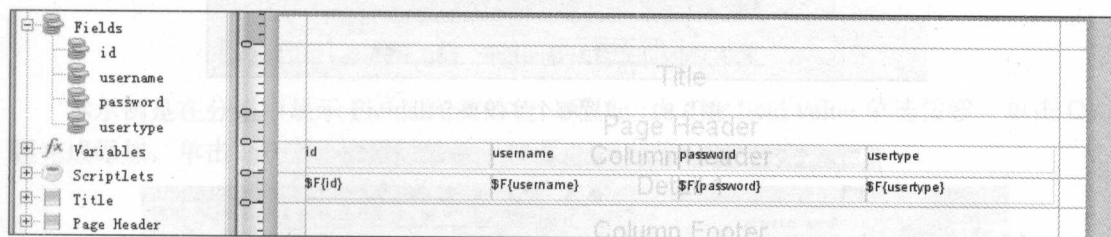


图 3.80 添加 4 个 Fields 对象

下面的步骤很关键，创建一个分组，即右键单击 report1 选项，弹出如图 3.81 所示的快捷菜单。

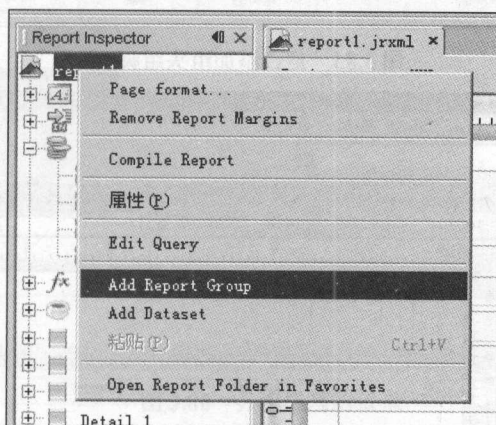


图 3.81 添加一个分组

选择 Add Report Group 命令后弹出如图 3.82 所示的对话框。设置完毕后单击“下一步”按钮继续配置，弹出如图 3.83 所示的对话框。设置完毕后单击“完成”按钮，分组 Group 添加完毕后的界面如图 3.84 所示。

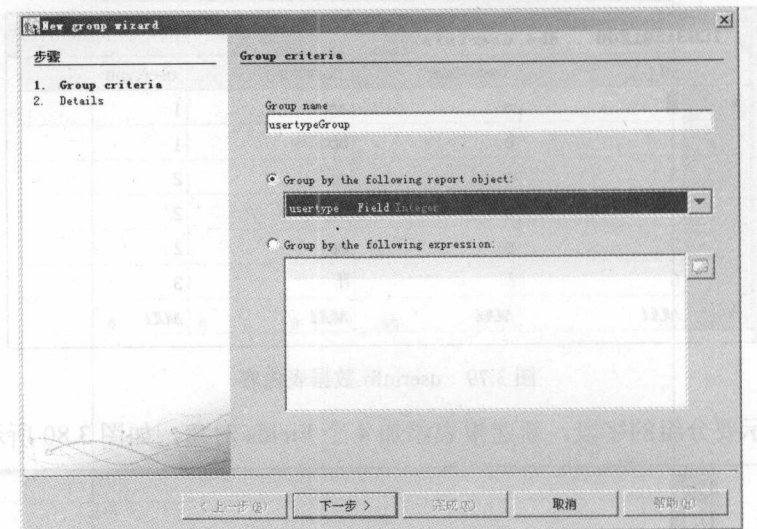


图 3.82 设置哪个字段要分组

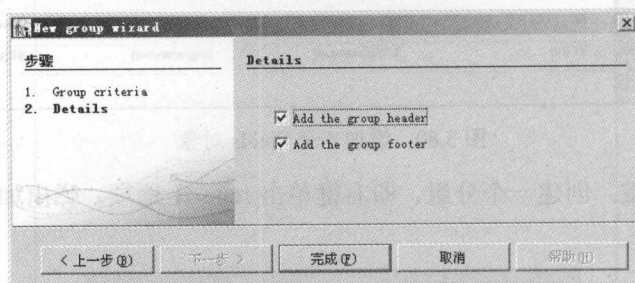


图 3.83 是否添加组头组脚

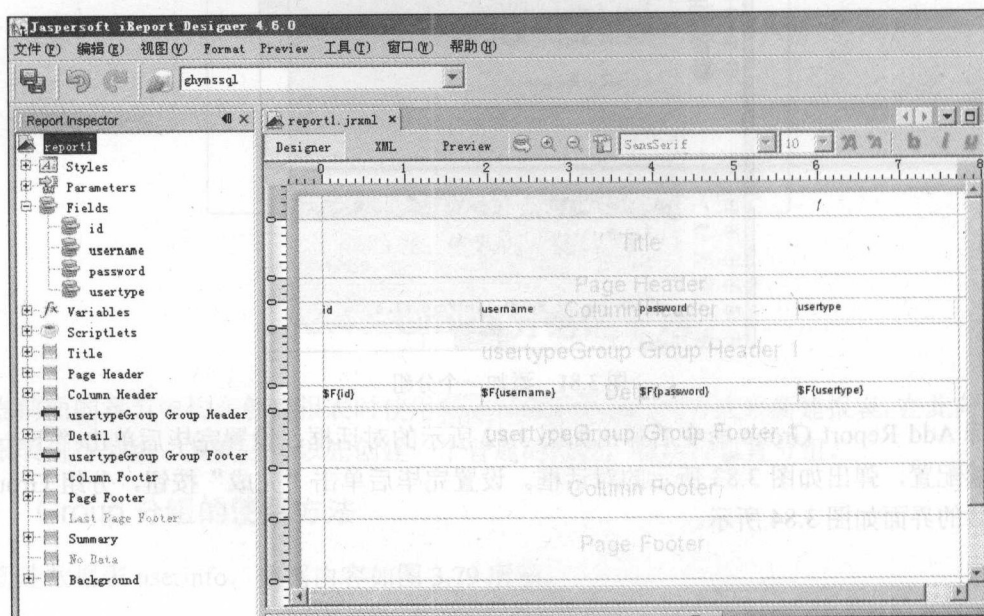


图 3.84 添加分组完毕后的界面

在图 3.84 中可以看到，新添加了两个 Band，一个是 usertypeGroup Group Header，另一个是 usertypeGroup Group Footer。

然后把名称为 usertype 的 Fields 对象拖曳到 usertypeGroup Group Header 栏中，弹出如图 3.85 所示的对话框。

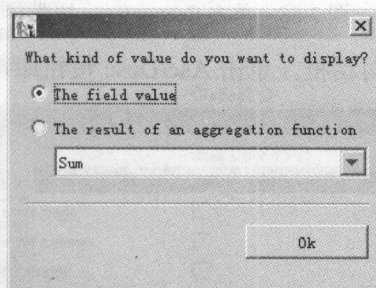


图 3.85 向 Band 中添加 Fields 对象

本示例是在分组中显示 Fields 当前的值，所以选中 The field value 单选按钮，单击 OK 按钮完成添加，单击 Preview 按钮后的效果如图 3.86 所示。

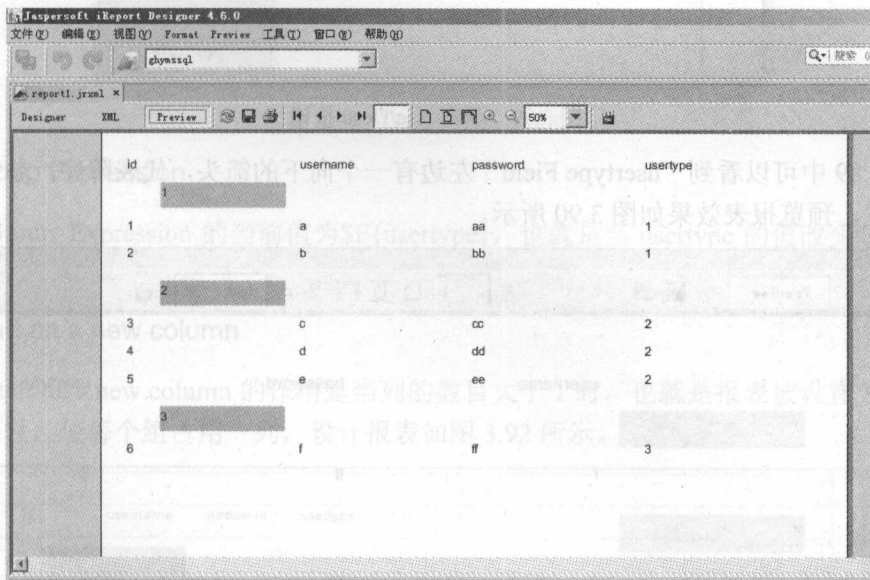


图 3.86 分组成功的效果

从图 3.86 中可以看到，根据 usertype 的值升序排序分组，那如何降序排序呢？单击 Report query 按钮配置查询条件，如图 3.87 所示。

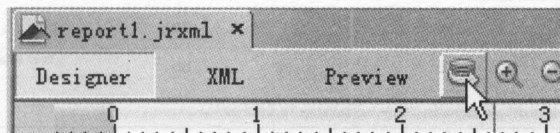


图 3.87 单击 Report query 按钮配置查询条件

在弹出的对话框中单击下方的 Sort options 按钮，如图 3.88 所示。

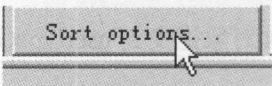


图 3.88 单击 Sort options 按钮

弹出如图 3.89 所示的设置对话框，单击 Asc/Dsc 按钮即可调整排序方向。

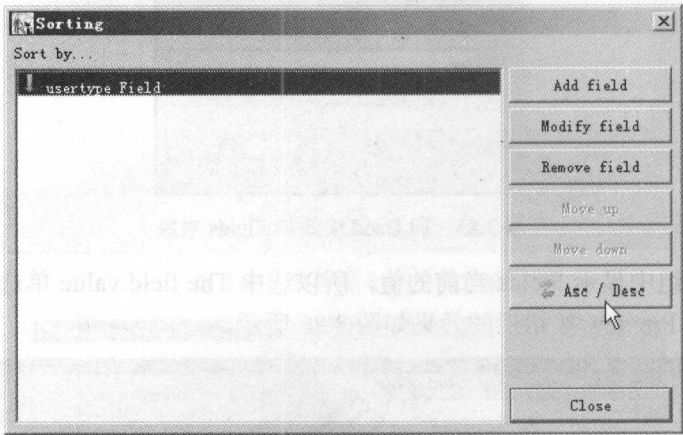
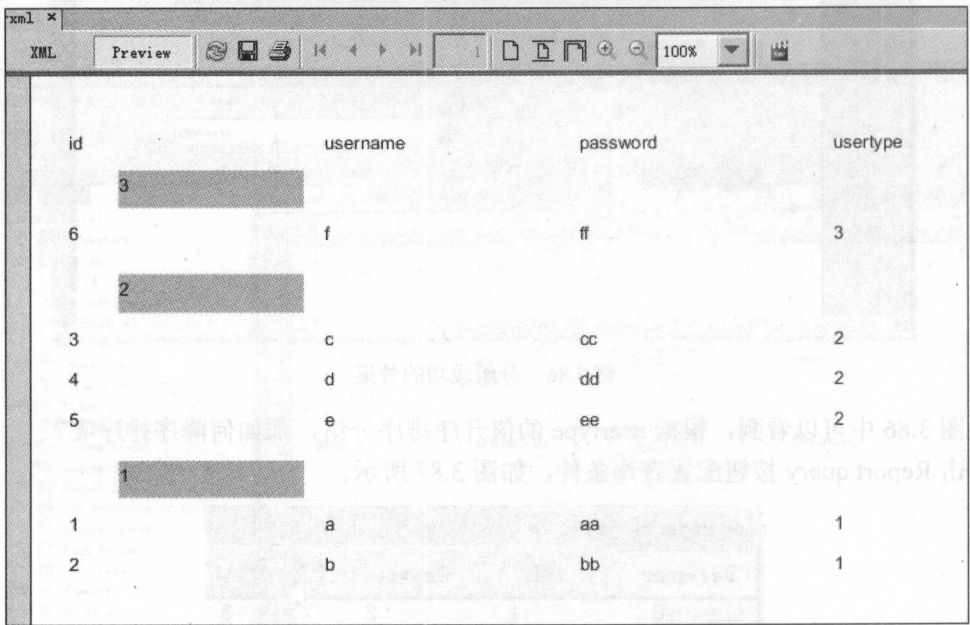


图 3.89 单击 Asc/Dsc 按钮

在图 3.89 中可以看到“usertype Field”左边有一个向下的箭头，代表降序，单击 Close 按钮完成配置，预览报表效果如图 3.90 所示。



id	username	password	usertype
3			3
6	f	ff	3
2			2
3	c	cc	2
4	d	dd	2
5	e	ee	2
1			1
1	a	aa	1
2	b	bb	1

图 3.90 降序排列

3.4.2 Group 分组的常用属性

分组 Group 也是一个 Band 对象，所以它也有一些属性，如图 3.91 所示。

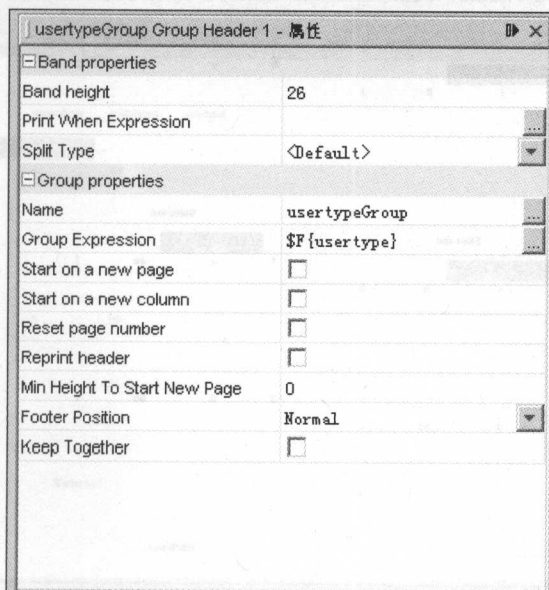


图 3.91 Group 分组的属性

1. Group Expression

属性 Group Expression 的当前值为 `$F{usertype}`，也就是当 `usertype` 的值改变时，新的组就被创建了。

2. Start on a new column

属性 Start on a new column 的作用是将列的数目大于 1 时，也就是报表被设置为多列时，则勾选该属性，使每个组占用一列，设计报表如图 3.92 所示。

Page Header			
id	username	password	usertype
Group Header			
\$F{usertype}			
usertypeGroup Group Header 1			
\$F{id}	\$F{username}	\$F{password}	\$F{usertype}
Detail 1			
Static text			
usertypeGroup Group Footer 1			
Column Footer			
Page Footer			

图 3.92 两列的 Group 分组报表

程序运行后的效果如图 3.93 所示。

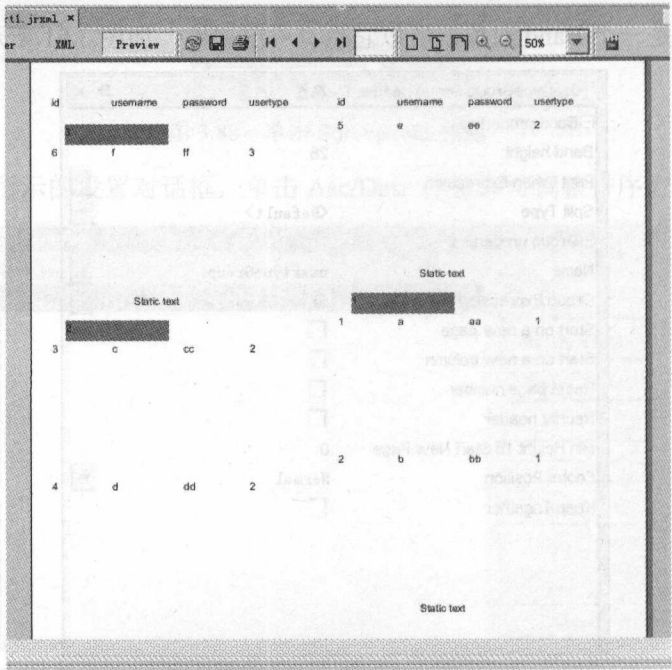


图 3.93 两列分组的运行效果

当勾选 Start on a new column 属性后，运行效果如图 3.94 所示。

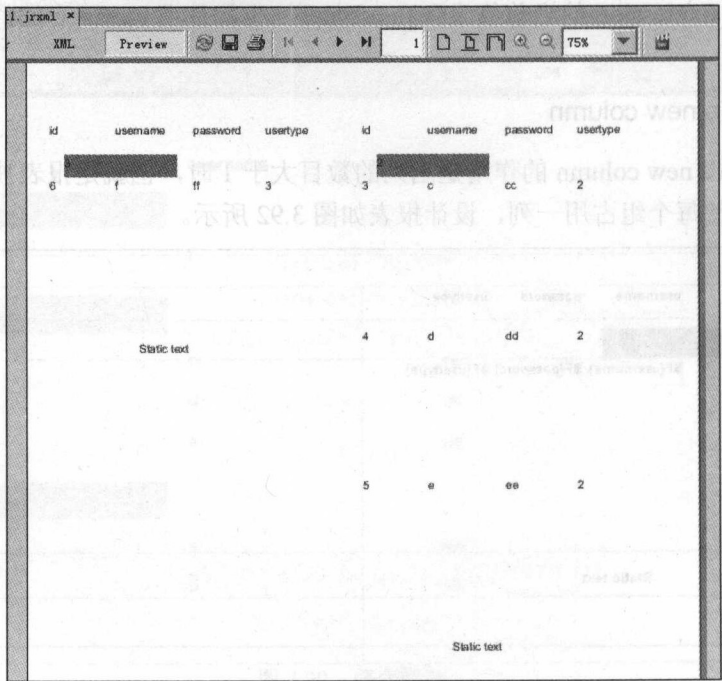
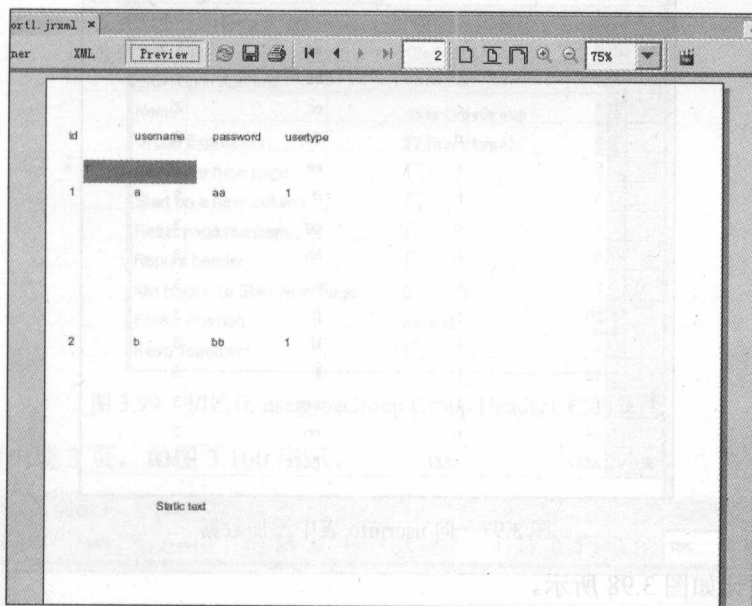


图 3.94 第 1 页运行效果

从图 3.94 中可以看到，每一组占用一列，继续看第 2 页是不是这样的效果，如图 3.95 所示。

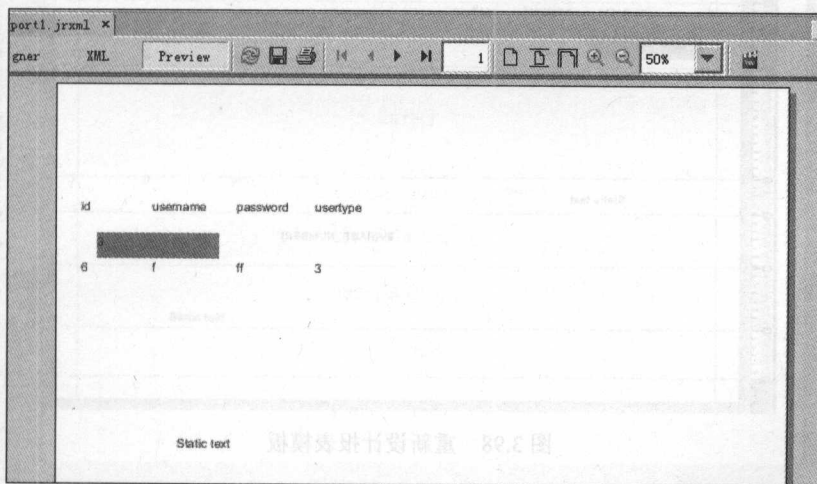


id	username	password	usertype
1	a	aa	1
2	b	bb	1

图 3.95 第 2 页运行效果

3. Start on a new page

属性 Start on a new page 用于是否将每一组打印在新的页中，还是使用上一个示例的报表模板，如图 3.92 所示，若选中 Start on a new page 属性，则程序运行后的效果如图 3.96 所示。



id	username	password	usertype
6	f	ff	3

图 3.96 每一组打印在一个单独的页面中

4. Reset page number

属性 Reset page number 是在每打印新的一组时将当前组的页码重置为 1，此示例需要数据

表中的数据很多，添加数据表 userinfo 的内容，如图 3.97 所示。

TC03\SQL200... dbo. userinfo				
	id	username	password	usertype
▶	1	a	aa	1
	2	b	bb	1
	3	c	cc	2
	4	d	dd	2
	5	e	ee	2
	6	f	ff	3
	7	g	gg	3
	8	h	hh	3
	9	i	ii	3
	10	j	jj	3
	11	k	kk	3
	12	l	ll	3
	13	m	mm	3
	14	n	nn	3
*	NULL	NULL	NULL	NULL

图 3.97 向 userinfo 表中添加数据

报表模板设计如图 3.98 所示。

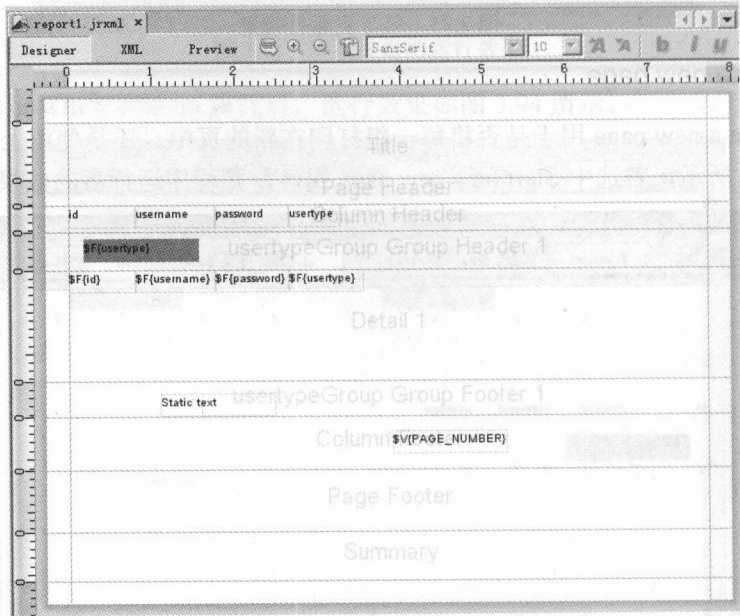


图 3.98 重新设计报表模板

在 Page Footer 中添加 `$V(PAGE_NUMBER)`（用于显示页数的 Tools 工具控件）。

设置报表的 Float column footer 属性为 true，还要设置 usertypeGroup Group Header1 栏的属性如图 3.99 所示。

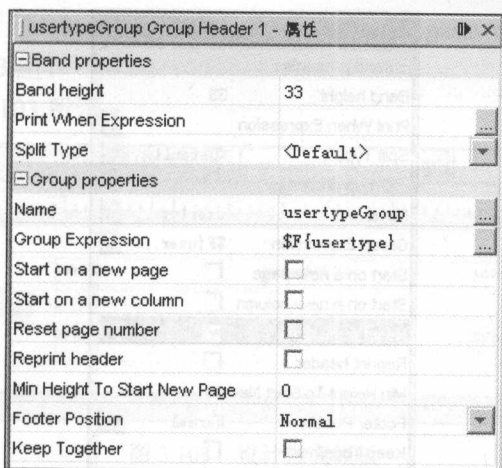


图 3.99 初始化 usertypeGroup Group Header1 栏的属性

程序运行后出现 3 页，如图 3.100 所示。

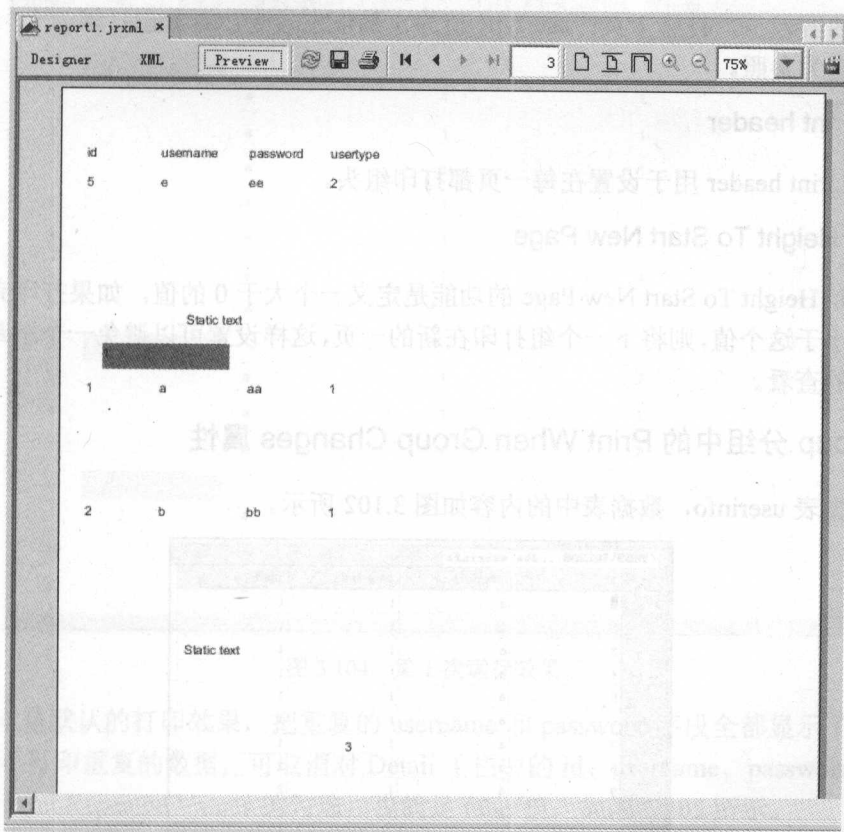


图 3.100 报表总页数为 3 页

然后，设置 usertypeGroup Group Header1 栏的 Reset page number 为 true，效果如图 3.101 所示。

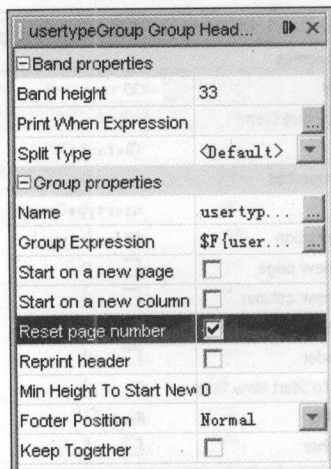


图 3.101 Reset page number 为 true

此时的运行效果是：usertype 中值为 3 的类别内容较多，所以打印出 1 页和 2 页；usertype 中值为 2 的内容较少，打印 1 页；usertype 值为 1 的值也较少，页数也打印为 1。也就是说页数是以当前组为参照。

5. Reprint header

属性 Reprint header 用于设置在每一页都打印组头。

6. Min Height To Start New Page

属性 Min Height To Start New Page 的功能是定义一个大于 0 的值，如果打印完某个组后，剩余的空间小于这个值，则将下一个组打印在新的一页，这样设置可以避免一个组跨越多个页，不便于报表的查看。

3.4.3 Group 分组中的 Print When Group Changes 属性

新建数据表 userinfo，数据表中的内容如图 3.102 所示。

TC03\SQL200...dbo.userinfo				
id	username	password	usertype	
1	a	1	1	
2	a	1	1	
3	a	1	1	
4	a	1	1	
5	a	1	1	
6	a	1	1	
7	a	1	1	
8	a	1	2	
9	a	1	2	
10	a	1	2	
11	a	1	3	
12	a	1	3	
13	a	1	3	
*	NULL	NULL	NULL	

图 3.102 userinfo 数据表中的内容

在图 3.102 中可以看到, userinfo 表中的 username 和 password 字段值完全相同, 只是 usertype 被分成了 1、2、3 组。

设计报表模板如图 3.103 所示。

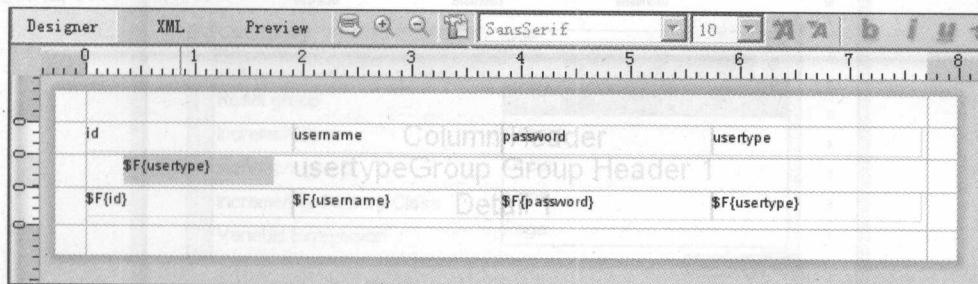


图 3.103 设计报表模板

报表模板运行后的效果如图 3.104 所示。

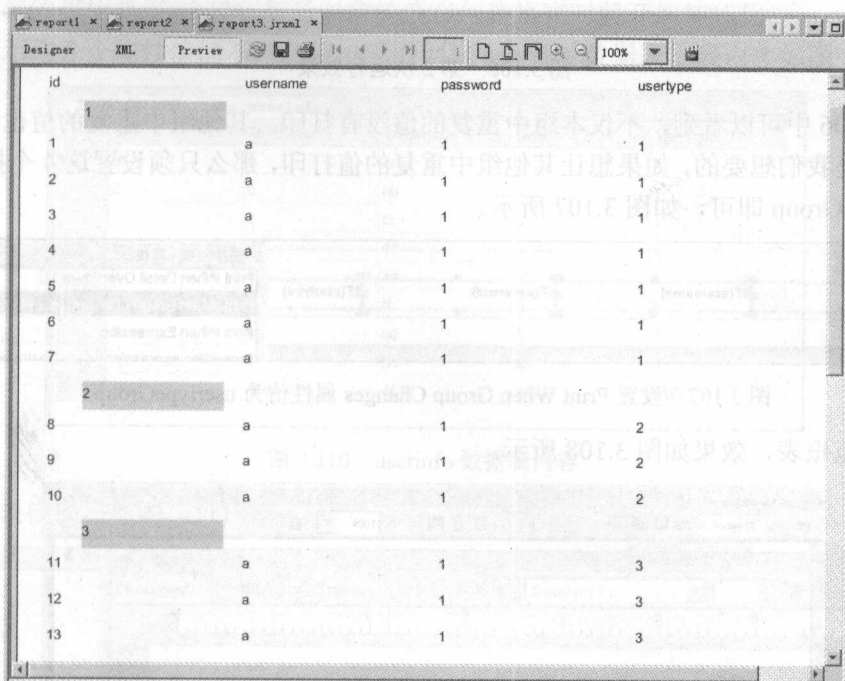


图 3.104 第 1 次运行效果

图 3.104 是默认的打印效果, 把重复的 username 和 password 字段全都显示了出来, 若想在本次示例中不打印重复的数据, 可取消对 Detail 1 栏中的 id、username、password 和 usertype 的控件属性 Print Repeated Values 的勾选, 也就是 false 值, 如图 3.105 所示。

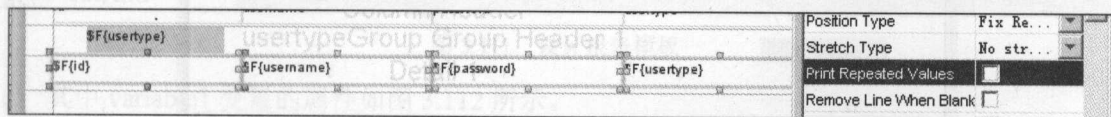
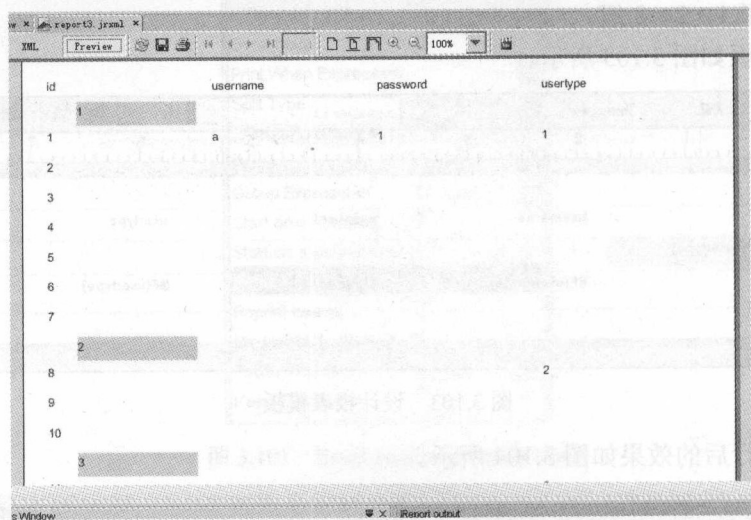


图 3.105 不勾选 Print Repeated Values

重新运行这个报表，效果如图 3.106 所示。



id	username	password	usertype
1	a	1	1
2	a	1	1
3	a	1	1
4	a	1	1
5	a	1	1
6	a	1	1
7	a	1	1
8	a	1	2
9	a	1	2
10	a	1	2

图 3.106 第 2 次运行效果

从图 3.106 中可以看到，不仅本组中重复的值没有打印，其他组中重复的值也没有打印，这种效果不是我们想要的，如果想让其他组中重复的值打印，那么只须设置这 4 个控件的属性值为 usertypeGroup 即可，如图 3.107 所示。

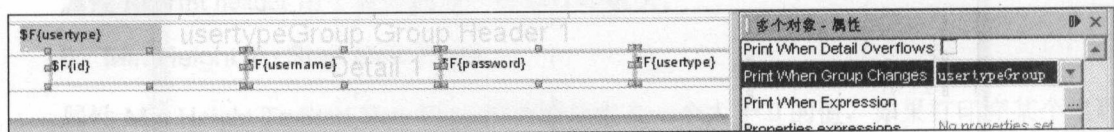
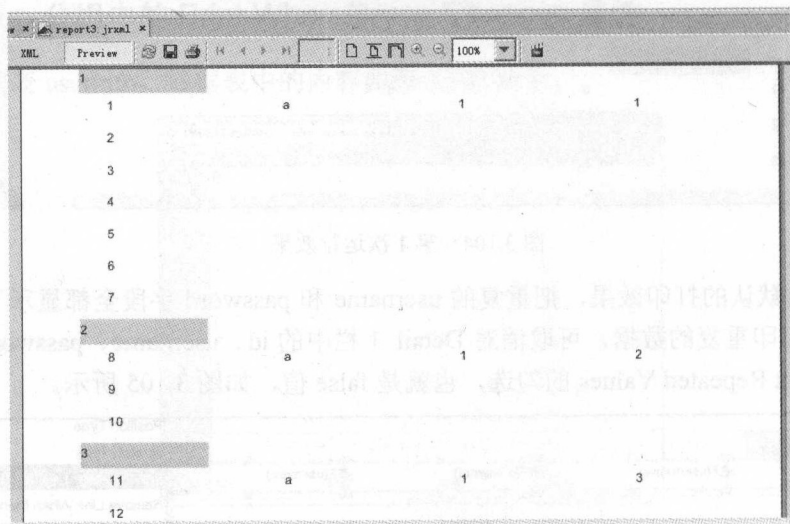


图 3.107 设置 Print When Group Changes 属性值为 usertypeGroup

再次预览报表，效果如图 3.108 所示。



id	username	password	usertype
1	a	1	1
2	a	1	1
3	a	1	1
4	a	1	1
5	a	1	1
6	a	1	1
7	a	1	1
8	a	1	2
9	a	1	2
10	a	1	2

图 3.108 最后一次预览效果

3.4.4 Group 分组中的 Reset type 属性

属性 Reset type 的主要作用就是重置 Variables 变量对象的值或决定赋值的时机。它的取值类型如图 3.109 所示。

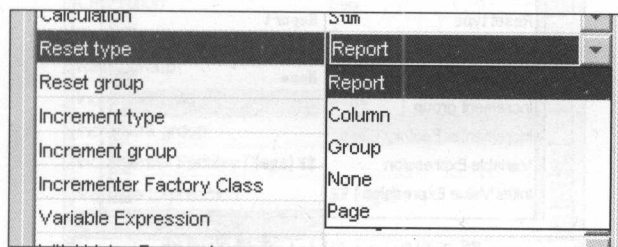


图 3.109 Reset type 属性的取值范围

1. Report 属性值

该属性值的作用是 Variables 变量对象的值在创建报表时就开始初始化。

为了演示这 5 个属性值的不同功能，下面创建 userinfo 数据表，表内容如图 3.110 所示。

TC05\SQL200... dba.userinfo					
	id	username	password	age	usertype
1	1	a	aa	2	1
2	2	b	bb	2	1
3	3	c	cc	1	1
4	4	d	dd	1	2
5	5	e	ee	2	2
6	6	f	ff	3	2
7	7	g	gg	1	3
8	8	h	hh	2	3
*	NULL	NULL	NULL	NULL	NULL

图 3.110 userinfo 数据表内容

创建报表模板，如图 3.111 所示。

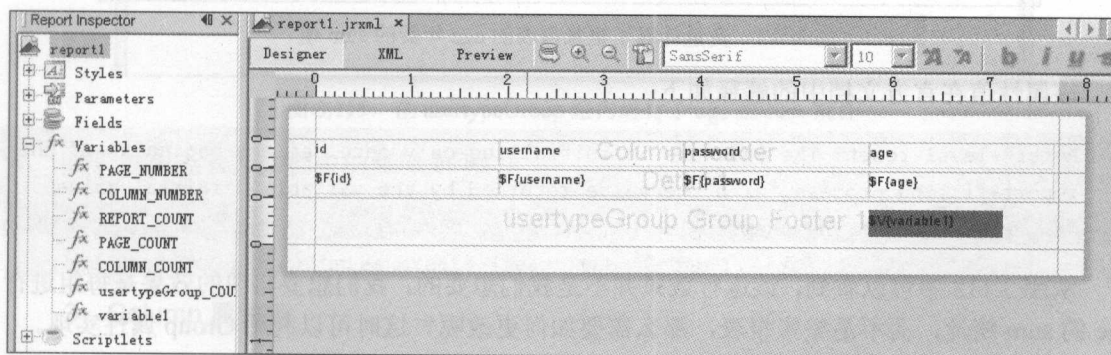


图 3.111 设计报表模板

其中 variable1 变量的属性如图 3.112 所示。

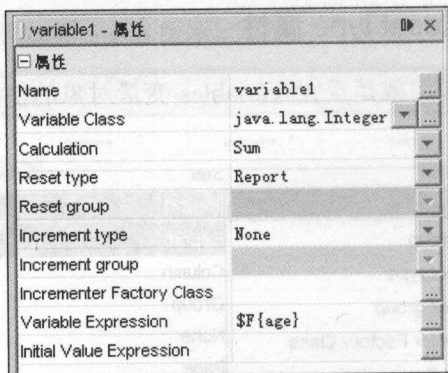


图 3.112 variable1 变量的属性

程序运行效果如图 3.113 所示。

id	username	password	age
1	a	aa	2
2	b	bb	2
3	c	cc	1
4	d	dd	1
5	e	ee	2
6	f	ff	3
7	g	gg	1
8	h	hh	2

图 3.113 变量计算了整张报表的 age 年龄的 sum 值

该属性值在官方文档中的解释如下：

Report-level reset: The variable is initialized only once, at the beginning of the report-filling process, with the value returned by the variable's initial value expression (resetType="Report").

从图 3.113 中可以看到，该运行效果并不是我们想要的，我们想要实现的效果是每组进行 age 的 sum 统计，而不是整张报表，那么需要如何更改呢？这时可以利用 Group 属性实现。

2. Group 属性值

属性值 Group 的作用是每个组内进行 Variables 变量对象的统计，当创建出新的 Group 组时，该变量的值被自动清 0，设置属性值如图 3.114 所示。

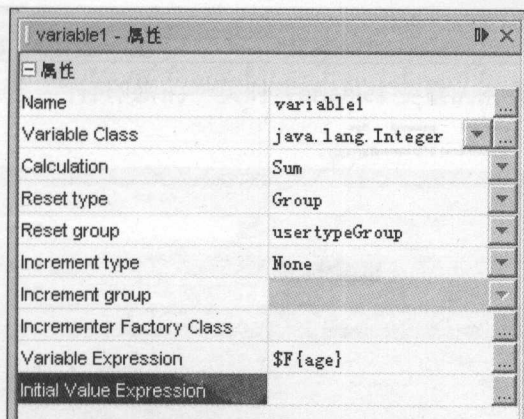


图 3.114 设置属性值为 Group

当 Reset type 属性值被设置为 Group 时, Reset group 属性值被自动设置为 usertypeGroup 组, 也就是要对 usertypeGroup 组中的 age 进行 sum 统计, 运行效果如图 3.115 所示。

id	username	password	age
1	a	aa	2
2	b	bb	2
3	c	cc	1
Sum			
4	d	dd	1
5	e	ee	2
6	f	ff	3
Sum			
7	g	gg	1
8	h	hh	2
Sum			

图 3.115 在 usertypeGroup 组内进行了 age 的 sum 统计

该属性值在官方文档中的解释如下:

Group-level reset: The variable is reinitialized every time the group specified by the resetGroup attributes breaks (resetType="Group").

3. Column 属性值

属性值为 Column 的作用是: 当以非 1 列的报表打印数据时, 如 2 列、3 列这样的报表模板, 在新的 column 列打印时变量 Variables 的值被重置。

设计报表如图 3.116 所示。

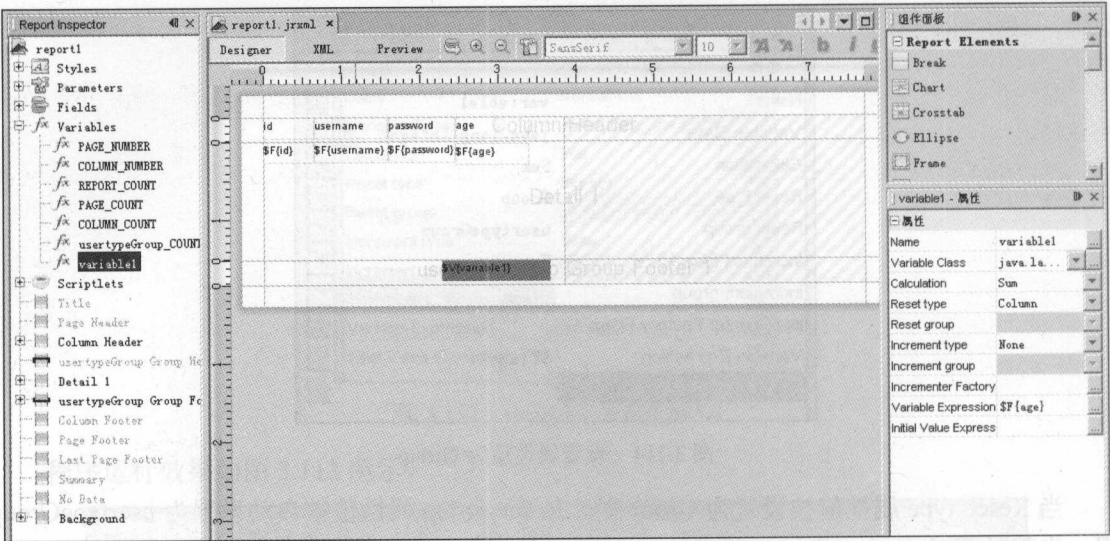


图 3.116 为 Column 重新设计报表模板

在图 3.116 中可以看到，报表模板有 2 列，并且设置变量 variable1 的 Reset type 属性值为 Column，报表运行后的效果如图 3.117 所示。

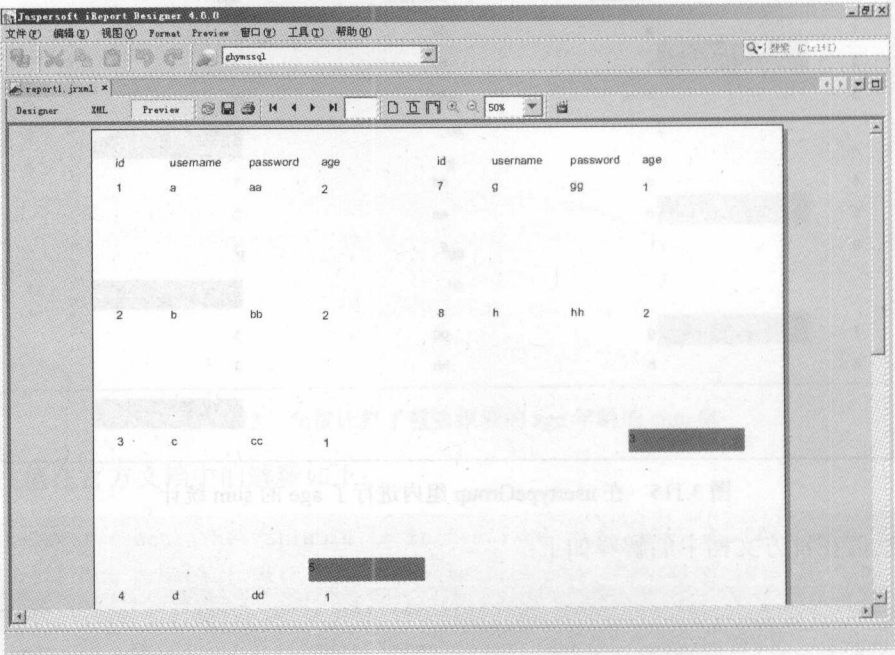


图 3.117 变量 variable1 在第 2 列打印时被重置

该属性值在官方文档中的解释如下：

Column-level reset: The variable is reinitialized at the beginning of each new column (resetType="Column").

4. Page 属性值

属性值 Page 的作用是：当打印新的一页时 Variables 变量的对象值被重置，设计报表模板如图 3.118 所示。

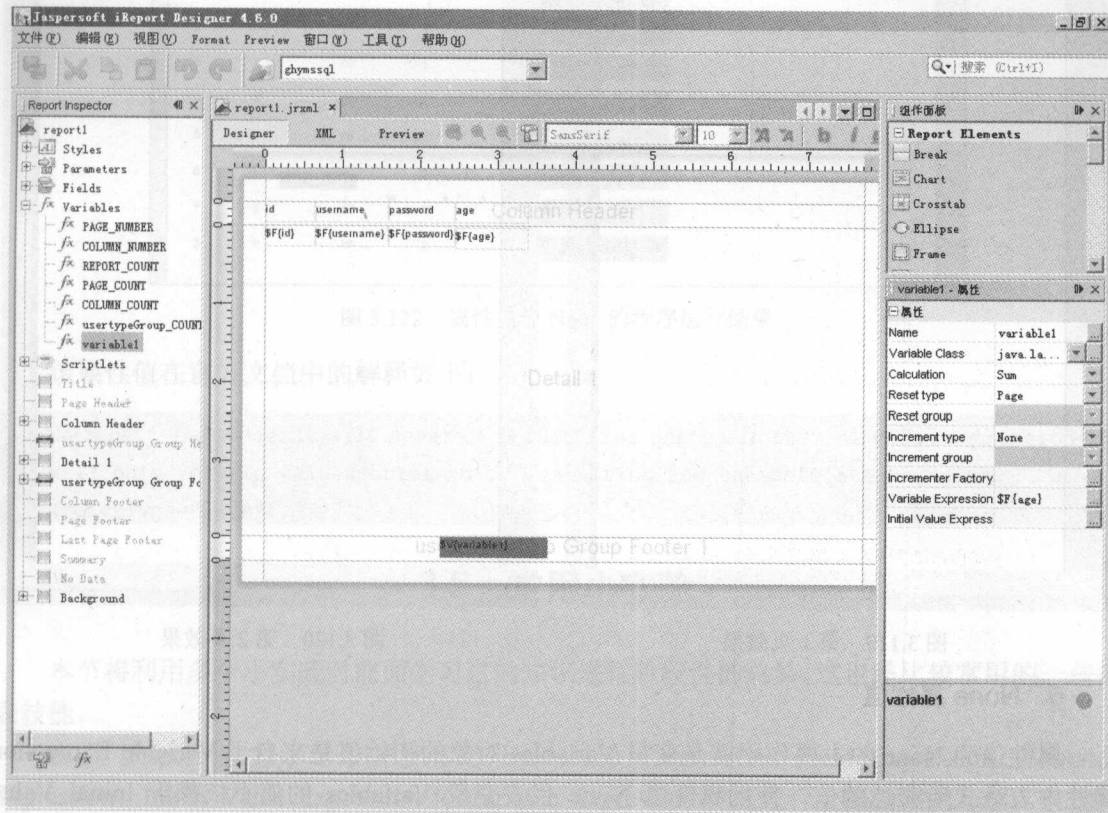


图 3.118 为 Page 重新设计的报表模板

程序运行后的第 1 页效果如图 3.119 所示。

从图 3.119 中并没有看到统计的 Sum 值，单击第 2 页看看，效果如图 3.120 所示。

从图 3.120 中可以看到，由于打印了新的一页，所以 Variables 变量被重置，结果为 1，这是正确的。

该属性值在官方文档中的解释如下：

Page-level reset: The variable is reinitialized at the beginning of each new page (resetType="Page").

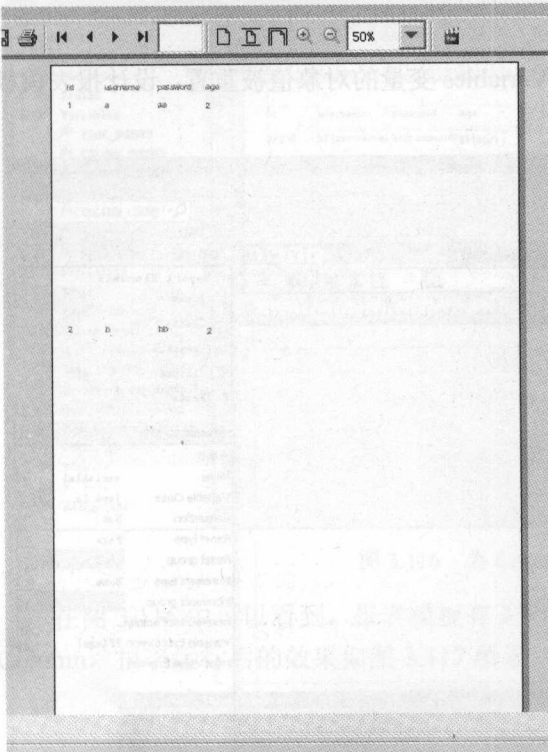


图 3.119 第 1 页效果

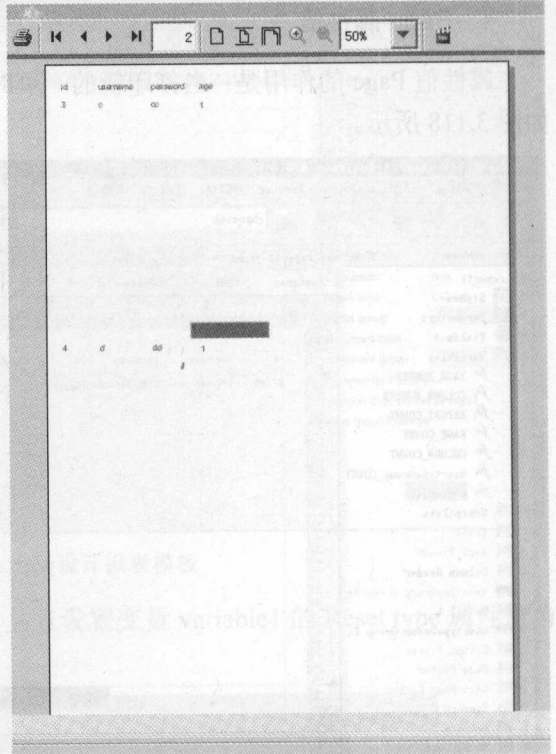


图 3.120 第 2 页效果

5. None 属性值

属性值为 None 的主要作用就是变量 Variables 对象的初始值是来自于 Variable Expression 属性中表达式所表达的值，使用属性值 None 代表变量 Variables 的值不依赖于 Initial Value Expression 属性。

设置报表模板，内容如图 3.121 所示。

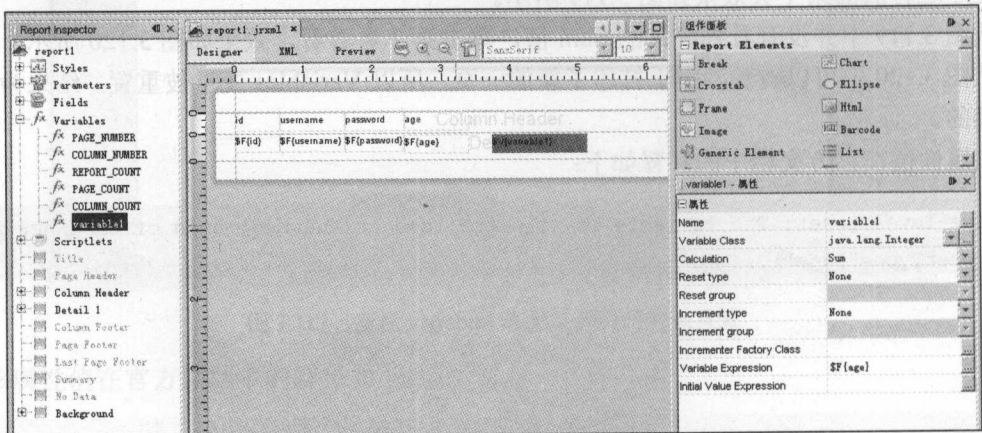


图 3.121 为 None 测试准备的模板报表

程序运行结果如图 3.122 所示。

id	username	password	age
1	a	aa	2
2	b	bb	2
3	c	cc	1
4	d	dd	1
5	e	ee	2
6	f	ff	3
7	g	gg	1
8	h	hh	2

图 3.122 属性值为 None 的程序运行结果

该属性值在官方文档中的解释如下：

No reset: The variable will never be initialized using its initial value expression and will only contain values obtained by evaluating the variable's expression (resetType="None").

3.5 常用小实验

本节将利用多个小实验对前面学习过的知识进行阶段性的总结,这也是比较常用的一些报表技能。

由于 iReport 官方手册中并没有对 Evaluation Time、Text Field Expression、Variable Expression、Initial Value Expression 和 Variable Expression 属性进行全面的案例介绍,仅仅以文字的方式介绍它们的功能,所以笔者针对这几个属性进行相应实验,以便为大家详细讲解在不同情况下的处理方式。

3.5.1 实验 1

数据表 userinfo 的内容如图 3.123 所示。

TC05\SQL200... dbo. userinfo		TC05\SQL200... dbo. userinfo		
	id	username	age	usertype
▶	1	a	2	1
	3	c	8	1
	4	d	1	2
	5	e	3	2
	6	f	5	2
	8	h	8	3
	9	i	5	4
*	NULL	NULL	NULL	NULL

图 3.123 userinfo 数据表内容

设计报表模板，内容如图 3.124 所示。

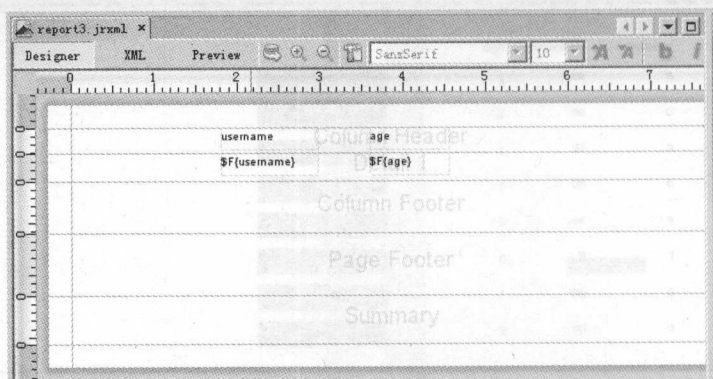


图 3.124 初始设计报表模板

程序运行后并没有序号从 1 开始的效果，所以要添加一个 Variables 变量对象，设置其属性如图 3.125 所示。

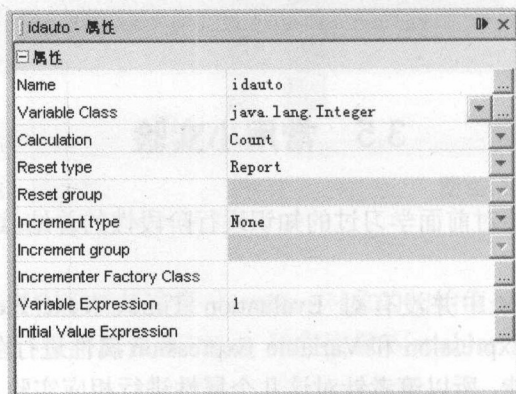


图 3.125 添加名称为 idauto 的 Variables 变量对象

程序运行后的效果如图 3.126 所示。

	username	age
1	a	2
2	c	8
3	d	1
4	e	3
5	f	5
6	h	8
7	i	5

图 3.126 id 自增

3.5.2 实验 2

新建报表并设计报表模板，然后添加一个 Variables 变量，其属性内容如图 3.127 所示。

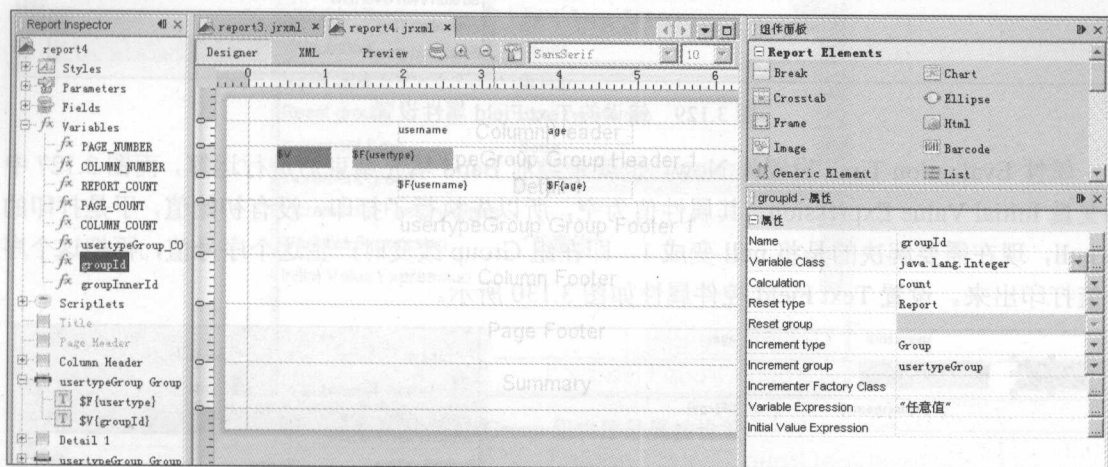


图 3.127 groupId 属性

该属性值在官方文档中的解释如下：

Group-level increment: The variable is incremented every time the group specified by the incrementGroup attributes breaks (incrementType="Group").

图 3.127 中属性设置的主要功能是在报表中计算“任意值”的个数，以每组为单位。程序运行后的效果如图 3.128 所示。

username	age
null	10
a	2
c	8
1	20
d	1
e	3
f	5

图 3.128 第 1 个位置为 null

出现这种效果的主要原因是由于 Text Field 控件的属性设置造成的，如图 3.129 所示。

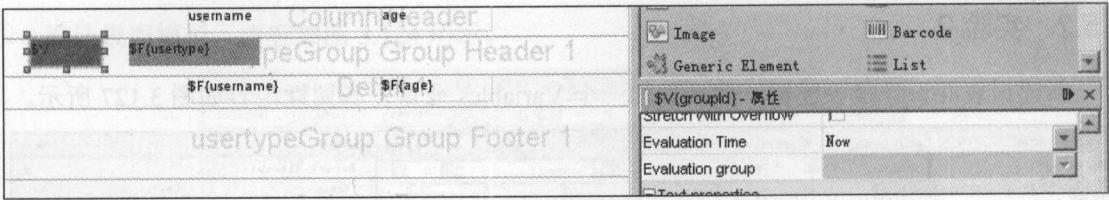


图 3.129 错误的 Text Field 属性设置

属性 Evaluation Time 设置为 Now，也就是当前 Band 填充结束后进行运算，而图 3.127 中的变量 Initial Value Expression，其属性值为空，所以先执行了打印，没有初始值，于是打印的是 null，现在需要解决的是将 null 变成 1，即在组 Group 改变时产生这个序号值，再把这个序号值打印出来。设置 Text Field 控件属性如图 3.130 所示。

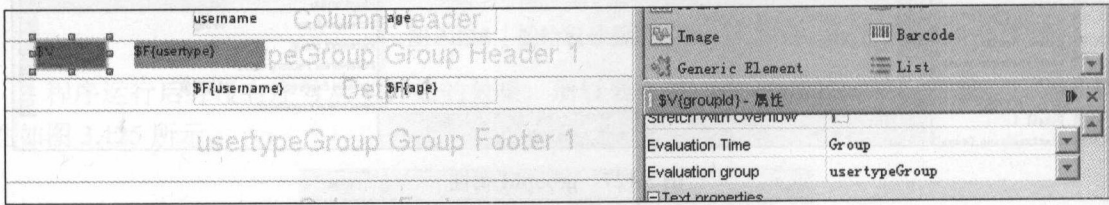


图 3.130 正确的 Text Field 属性设置

运行效果如图 3.131 所示。

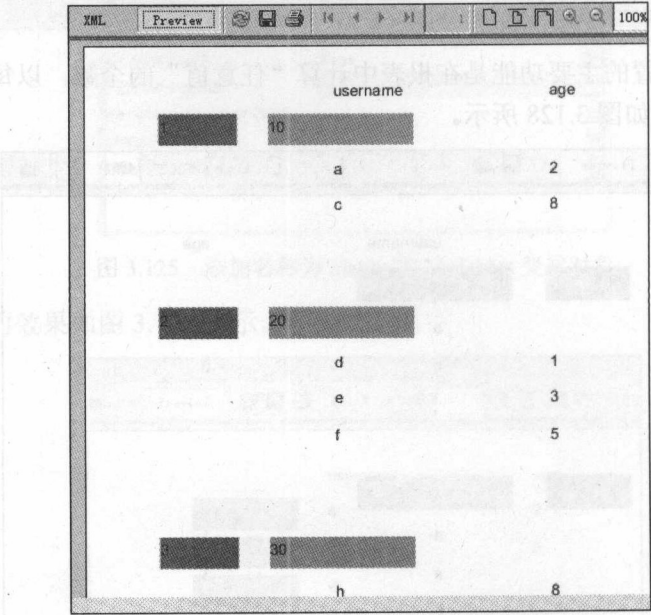


图 3.131 每组都有序号

但每一组内的序号却没有，继续添加 Variables 变量，属性设置如图 3.132 所示。

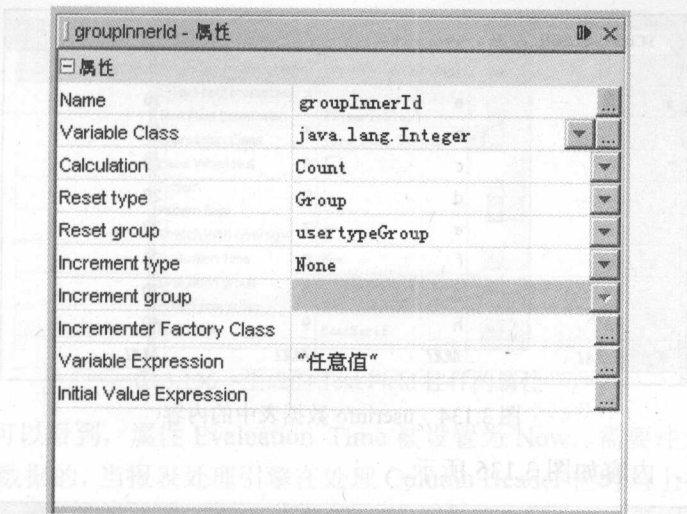


图 3.132 Group 组内序号属性设置

程序运行后的效果如图 3.133 所示。

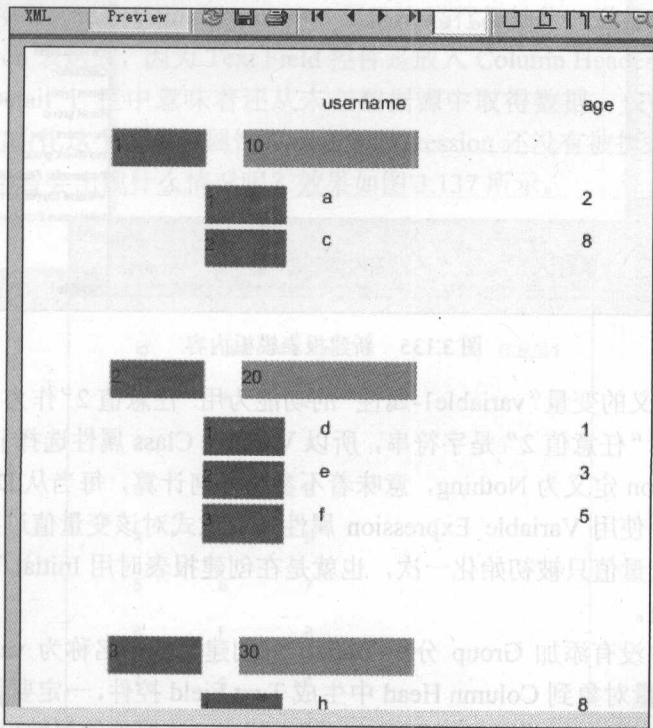


图 3.133 组内记录也有序号

3.5.3 实验 3

新建测试用的 userinfo 数据表，内容如图 3.134 所示。

TC03\SQL200... dba.userinfo				
	id	username	age	usertype
▶	1	a	2	10
	2	b	4	10
	3	c	6	10
	4	d	5	20
	5	e	7	20
	6	f	8	30
	7	g	4	30
	8	h	9	40
*	NULL	NULL	NULL	NULL

图 3.134 userinfo 数据表中的内容

新建报表模板，内容如图 3.135 所示。

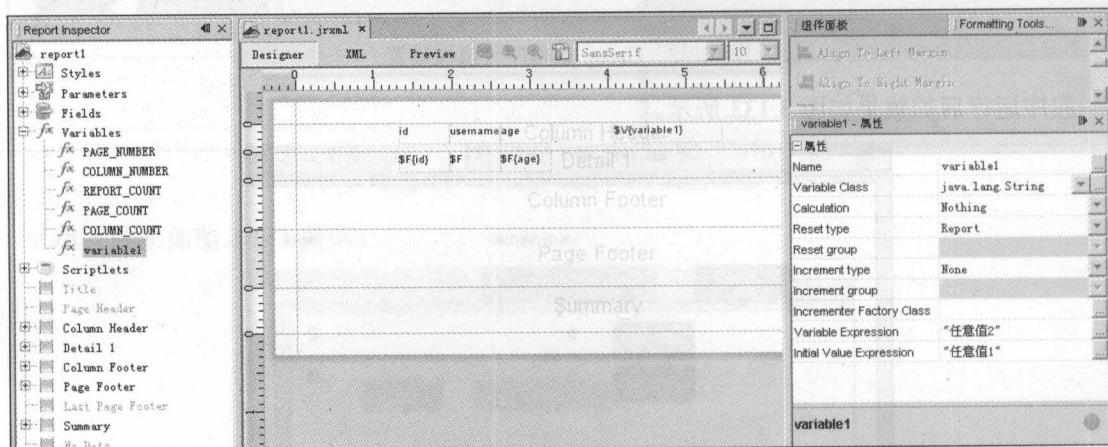


图 3.135 新建报表模板内容

在图 3.135 中定义的变量“variable1-属性”的功能为用“任意值 2”作为 Variable Expression 属性的表达式，由于“任意值 2”是字符串，所以 Variable Class 属性选择 java.lang.String 数据类型，属性 Calculation 定义为 Nothing，意味着不参与任何计算，每当从 DataSource 数据源读出一条新的记录时，使用 Variable Expression 属性的表达式对该变量值进行更新。Reset type 选择 Report，代表变量值只被初始化一次，也就是在创建报表时用 Initial Value Expression 属性初始化当前变量值。

另外在报表中并没有添加 Group 分组 Band，只创建了一个名称为 variable1 的 Variables 变量对象，拖动该变量对象到 Column Head 中生成 Text Field 控件，一定要留意的是：Text Field 控件被拖曳到 Column Header 栏中，并不是 Detail 1 栏中，该控件的属性设置如图 3.136 所示。

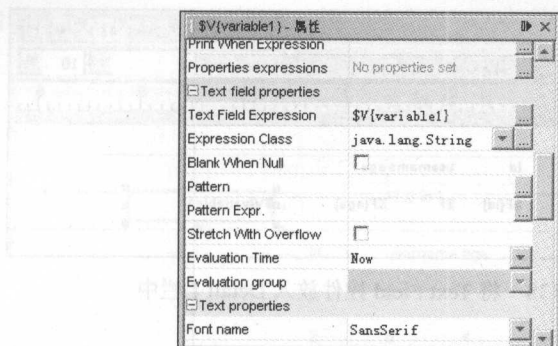


图 3.136 生成的 Text Field 控件的属性

从图 3.136 中可以看到，属性 Evaluation Time 被设置为 Now，-需要注意的是，处理报表时是从上到下填充数据的，当报表处理引擎在处理 Column Header 栏时马上打印这个 variable1 变量值，那么以什么样的计算方式打印这个值呢？其实在图 3.135 中已经有了答案，在该图中已经设置了 variable1 变量的 Calculation 属性值为 Nothing，也就是 variables1 变量的 Variable Expression 属性不参与任何类型的计算，并且它的 Reset type 属性被设置为 Report，也就是报表刚刚创建时就要使用 Initial Value Expression 属性来进行初始化，进行值的重置，但并不执行 Variable Expression 表达式，因为 Text Field 控件是放入 Column Header 栏中，而不是 Detail 1 栏中。不放在 Detail 1 栏中意味着还从未在数据源中取得数据，仅仅使用 Initial Value Expression 属性来初始化这个变量，属性 Variable Expression 还没有被执行到。

预览这个报表看看会出现什么情况呢？效果如图 3.137 所示。

id	username	age	任意值1
1	a	2	
2	b	4	
3	c	6	
4	d	5	
5	e	7	
6	f	8	
7	g	4	
8	h	9	

图 3.137 第 1 次预览效果

再重新设计报表模板，内容如图 3.138 所示。

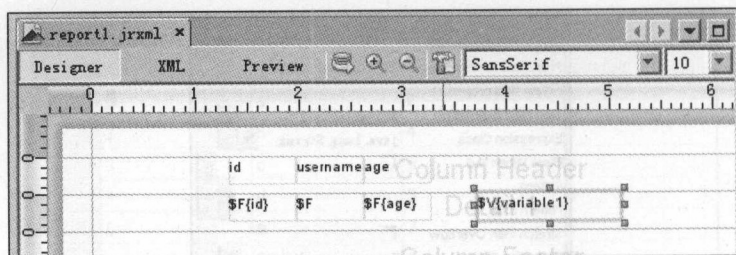


图 3.138 将 Text Field 控件放入 Detail 1 栏中

预览效果如图 3.139 所示。

id	username	age
1	a	2
2	b	4
3	c	6
4	d	5
5	e	7
6	f	8
7	g	4
8	h	9

图 3.139 第 2 次预览效果

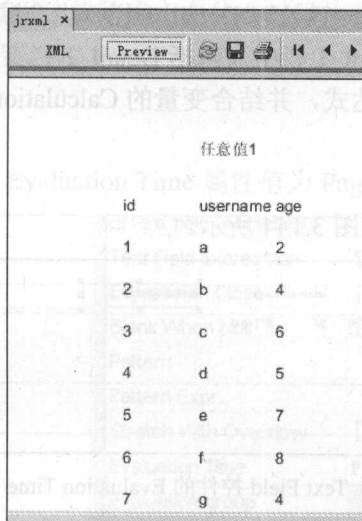
通过上面的实验可以得出一个过程：当 Text Field 控件的属性 Evaluation Time 值被设置为 Now 时，也就是立即执行 Text Field Expression 属性内的表达式，并且 Text Field 控件被放置在 Detail 1 栏上方某一个 Band 栏中，则只打印 Initial Value Expression 属性值，如果放入 Detail 1 栏中，则 Initial Value Expression 属性先执行，Variable Expression 后执行，所以最终的值也是 Variable Expression 属性值的计算结果。

放入 Title 栏中的效果也是相同的，因为 Title 栏的位置也是在 Detail 1 栏的上方，设计报表模板，内容如图 3.140 所示。

Title		
id	username	age
\$F{id}	\$F	\$F{age}

图 3.140 将 Text Field 添加到 Title 栏中

运行效果如图 3.141 所示。



id	username	age
1	a	2
2	b	4
3	c	6
4	d	5
5	e	7
6	f	8
7	g	4

图 3.141 在 Title 栏中显示“任意值 1”

那如果放在 Detail 1 栏的下方会出现什么样的效果呢？将 Text Field 控件放在 Detail 1 栏的下方，报表模板界面如图 3.142 所示。

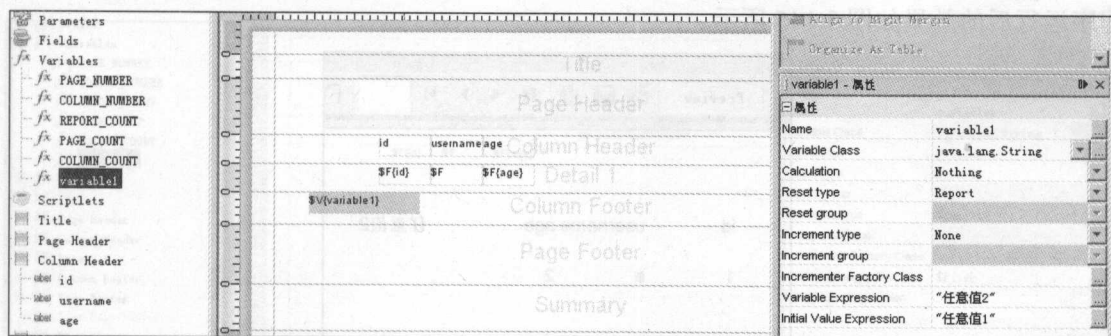
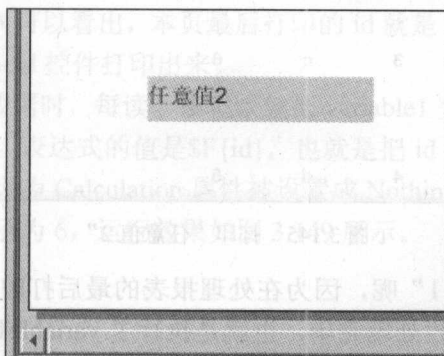


图 3.142 放在 Detail 1 栏下方的 Text Field 控件

运行效果如图 3.143 所示。



任意值2

图 3.143 打印任意值 2

上面的示例是以 Detail 1 栏为“楚河汉界”，分为上下结构，报表引擎从上到下处理 Variables 变量对象是不相同的，因为引擎一旦处理 Detail 1 栏，也会顺便把 Variables 对象处理了，所以如果 Text Field 控件放入 Detail 1 栏中（包括 Detail 1 栏下方的其他 Band 栏），都将执行 Variable Expression 属性中的表达式，并结合变量的 Calculation 计算类型，得出值后再打印。

3.5.4 实验 4

重新设计报表模板，内容如图 3.144 所示。

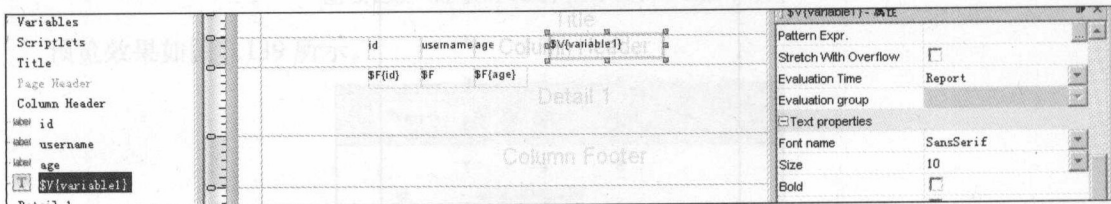


图 3.144 设计 Text Field 控件的 Evaluation Time 属性为 Report

其中 Report 属性值在官方文档中的解释为：

Report	Evaluate the expression at the end of the report
--------	--

也就是在处理完报表后，在最后阶段执行 Text Field 控件的 Text Field Expression 表达式，程序运行后的效果如图 3.145 所示。

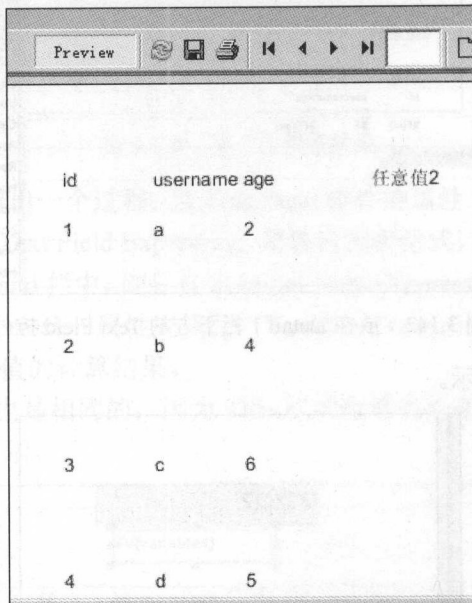


图 3.145 打印“任意值 2”

那为什么不是“任意值 1”呢，因为在处理报表的最后打印该值，在这个过程中报表引擎已经从上到下将 Detail 1 栏处理完毕，也就是执行了 variable1 变量的 Variable Expression 属性值所对应的表达式“任意值 2”，变量 variable1 被赋值，所以“任意值 2”就是最终打

印的结果。

Evaluation Time 属性的主要用途就是确定在什么时候执行 Text Field Expression 表达式, 该属性在使用 iReport 时非常重要。

3.5.5 实验 5

继续设置 Text Field 控件的 Evaluation Time 属性值为 Page, 效果如图 3.146 所示。

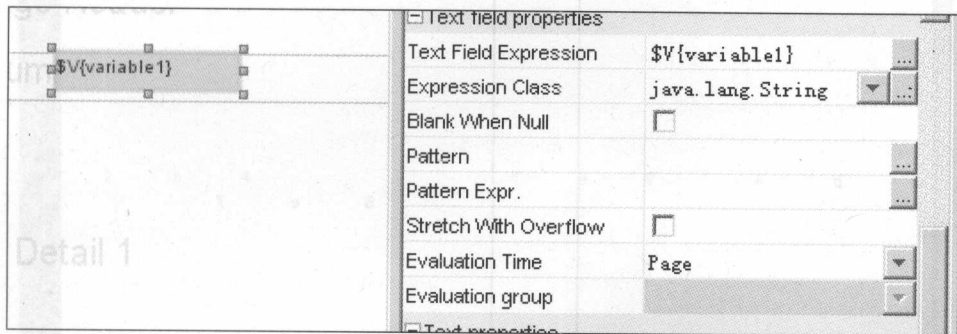


图 3.146 Evaluation Time 属性值为 Page

设置为 Page 属性值就意味着在本页结束时打印 variable1 变量的值, 其中 variable1 的属性设置如图 3.147 所示。

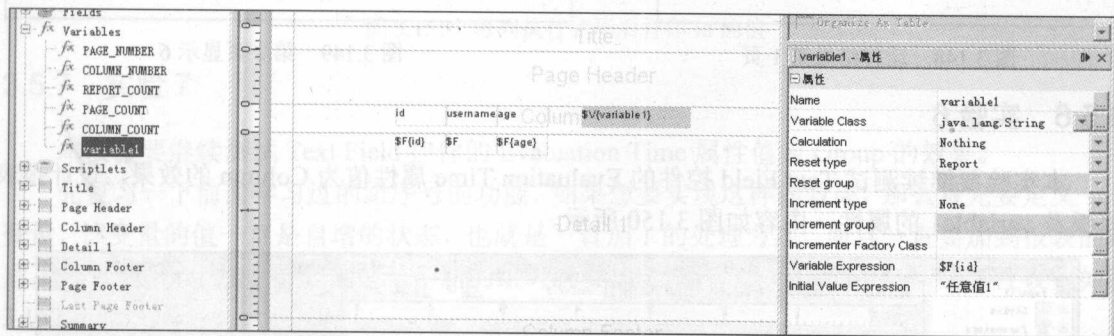


图 3.147 属性设置

在图 3.147 中将 Detail 1 栏加高, 程序预览后的效果如图 3.148 所示。

从图 3.148 中打印的值 3 可以看出, 本页最后打印的 id 就是 3, 所以在处理新的页面前要把 variable1 变量通过 Text Field 控件打印出来。

在处理 Detail 1 栏中的数据时, 每读一条记录就把 variable1 变量的 Variable Expression 属性值对应的表达式执行一次, 表达式的值是 \$F{id}, 也就是把 id 的值赋给 variable1 变量, 并且无任何的附加计算方式, 因为 Calculation 属性被设置成 Nothing 了。

依次类推, 第 2 页应显示为 6, 运行效果如图 3.149 所示。

id	username	age
1	a	2
2	b	4
3	c	6

图 3.148 运行结果第 1 页

id	username	age
4	d	5
5	e	7
6	f	8

图 3.149 第 2 页显示 6

3.5.6 实验 6

本实验要继续测试 Text Field 控件的 Evaluation Time 属性值为 Column 的效果，设计报表模板及 variable1 的属性，内容如图 3.150 所示。

Figure 3.150 shows the iReport Designer interface. The central area displays a report template with a table structure. The table has columns for id, username, and age. The table is divided into sections: Page Header, Column Header, Detail, Column Footer, Page Footer, and Summary. The 'variable1' property is highlighted in the table. The right-hand pane shows the 'variable1' properties, including Name, Variable Class, Calculation, Reset type, Reset group, Increment type, Increment group, Incrementer Factory Class, Variable Expression, and Initial Value Expression.

图 3.150 双列的报表模板

程序运行后的效果如图 3.151 所示。

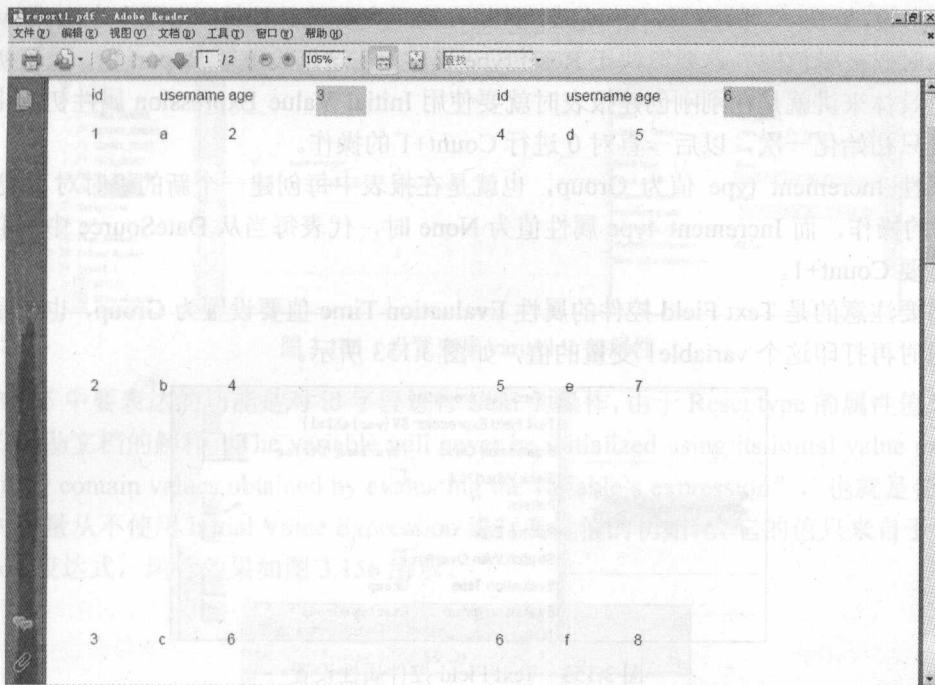


图 3.151 每列执行完毕后打印 id 的值

3.5.7 实验 7

本实验要继续测试 Text Field 控件的 Evaluation Time 属性值为 Group 的效果。

先复习一下前面学习过的组序号的功能，如果想要实现这样的效果，那么首先要定义一个变量，该变量的值一直是自增的状态，也就是一直加 1 的处理方式，而且一直要加到报表的最后，在什么情况下加 1 呢？创建一个新的组时就要加 1，所以该 variable1 变量的属性设置如图 3.152 所示。

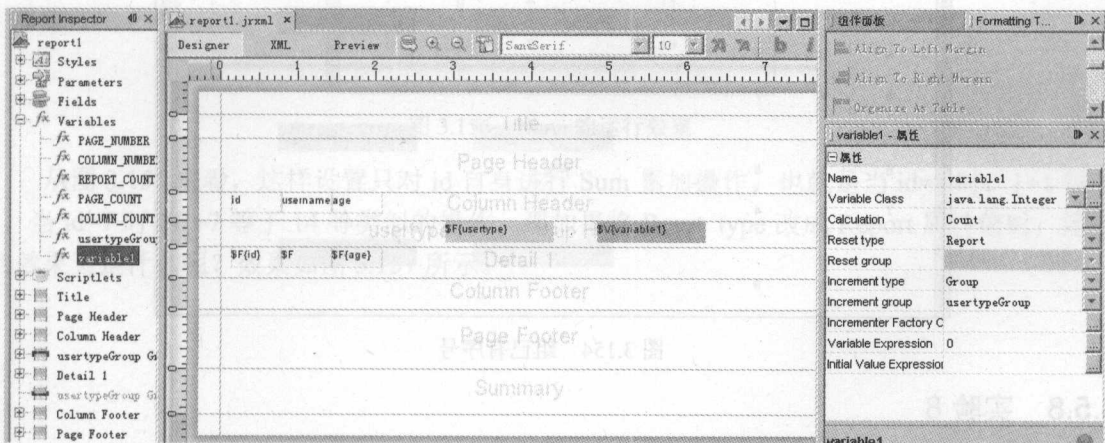


图 3.152 设置变量属性

在图 3.152 中设置变量的 Calculation 属性值为 Count, 也就是要计算 Variable Expression 属性 0 的个数, 值 0 可以随意写, 任何值都可以, 在这里主要就是计算 0 出现多少次。那 variable1 变量什么时候被赋值呢? 这要取决于 Reset type 属性, 在这里设置了 Report 值, 重置初始值的级别, 更具体来讲就是在刚刚创建报表时就要使用 Initial Value Expression 属性初始化变量的值, 并且只初始化一次, 以后一直对 0 进行 Count+1 的操作。

而属性 Increment type 值为 Group, 也就是在报表中每创建一个新的组时对 0 进行计数 Count+1 的操作, 而 Increment type 属性值为 None 时, 代表每当从 DataSource 中读取出一条记录时就要 Count+1。

还需要注意的是 Text Field 控件的属性 Evaluation Time 值要设置为 Group, 也就是当创建一个组时再打印这个 variable1 变量的值, 如图 3.153 所示。

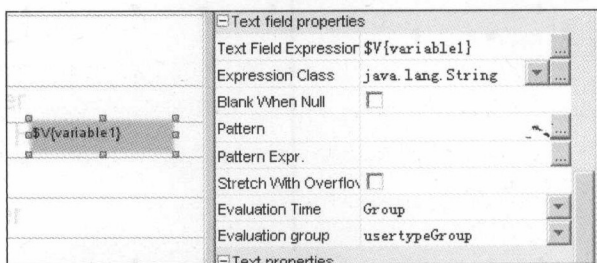


图 3.153 Text Field 控件属性设置

运行结果如图 3.154 所示。

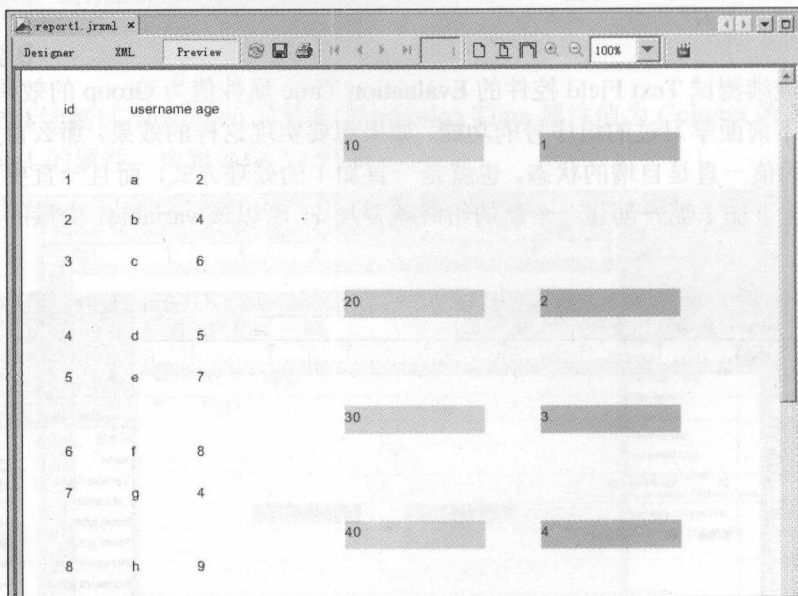


图 3.154 组已有序号

3.5.8 实验 8

前面已经学习了 Evaluation Time 属性的使用方法, 下面再研究一下 Reset type 属性值为

None 时的使用效果，如图 3.155 所示。

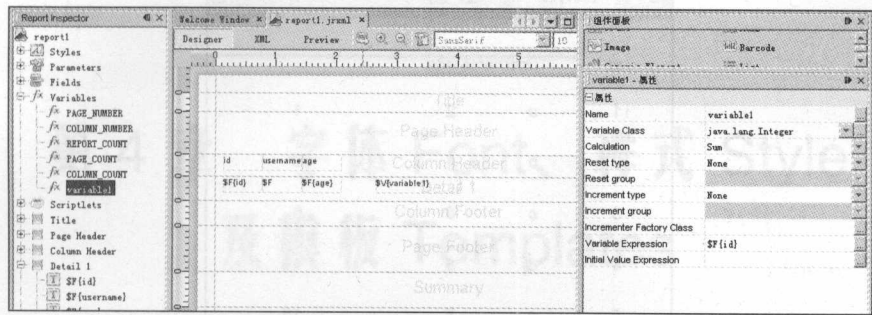


图 3.155 设置变量 variable1 的属性

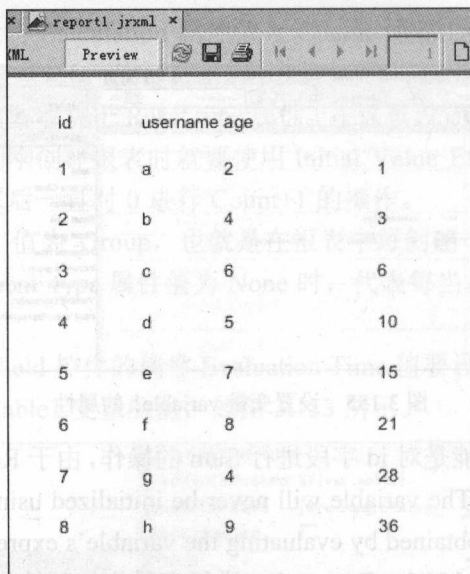
图 3.155 中要表达的功能是对 id 字段进行 Sum 的操作，由于 Reset type 的属性值为 None，根据官方帮助文档的解释 “The variable will never be initialized using its initial value expression and will only contain values obtained by evaluating the variable’s expression”，也就是使用 None 属性值时变量从不使用 Initial Value Expression 进行变量值的初始化，它的值只来自于 Variable Expression 表达式。运行效果如图 3.156 所示。

The screenshot shows the 'Report Preview' window. It displays a table with 4 columns: 'id', 'username', 'age', and an unlabeled column showing the sum of 'id'. The data is as follows:

id	username	age	
1	a	2	2
2	b	4	4
3	c	6	6
4	d	5	8
5	e	7	10
6	f	8	12
7	g	4	14
8	h	9	16

图 3.156 None 的运行效果

从图 3.156 来看，这样设置只对 id 自身进行 Sum 累加操作，也就是当 id=1 时，1+1 等于 2，当 id=7 时，7+7 等于 14 等类似的操作，那如果将 Reset type 改成 Report 属性值时，运行的效果又是什么呢？效果如图 3.157 所示。



id	username	age	
1	a	2	1
2	b	4	3
3	c	6	6
4	d	5	10
5	e	7	15
6	f	8	21
7	g	4	28
8	h	9	36

图 3.157 对 id 值进行累加的属性设置运行效果

通过本实验可以知道 Reset type 值为 None 和 Report 时的区别，一定要注意区分：属性值取 None 只对 Variable Expression 表达式进行计算，使用 Report 时要对整个报表中的 Variable Expression 表达式进行计算。

笔者经过测试得出以下经验。

- 当变量 Variables 的 Calculation 属性设置为 Nothing 时，代表变量不参与运算，这时 Reset type 属性值的作用被完全忽略，因为不参与运算也就没有重置的意义，打印出来的数据仅仅是 Variable Expression 表达式中的值。
- Calculation 属性值设置为非 Nothing 时，要对变量进行计算，这时 Reset type 设置为 None 时就意味着 Variable Expression 属性中的表达式从不进行重置，只使用 Variable Expression 表达式中的值进行指定类型的计算。
- Calculation 属性值设置为非 Nothing 时，并且 Reset type 也设置为非 None 时，代表变量既参与运算，又具有重置的功能。

3.5.8 实验 8

前面已经学习了 Evaluation Function 属性的设置方法，下面来学习一下 Reset type 属性值为

第 4 章 字体 Font、样式 Style 及模板 Templates



导言

报表也需要美化，本章将利用报表中的 Font 字体、Style 样式及 Templates 模板对报表进行美化，从而体现出报表设计的专业性，读者应该着重掌握如下内容：

- ★ 创建字体扩展
- ★ 创建样式 Style
- ★ 创建及使用报表模板 Templates

4.1 字体 Font

在 iReport 中支持使用当前操作系统自带的 TrueType 字体文件，如果觉得 iReport 中的字体文件比较少，可以将.ttf 字体文件放入 Windows 操作系统的 C:\WINDOWS\Fonts 路径下，还可以在 iReport 软件中用字体扩展功能把字体文件.ttf 导出为.jar 文件，并在“工具”菜单中的 Option 选项下的 classpath 列表中添加这个.jar 文件。

如果使用字体扩展功能，还可以把这些字体.jar 文件在导出的 PDF 文件中使用，也就是在导出的 PDF 文件中使用第三方的字体样式。

在 iReport 中设置字体比较简单，设置界面如图 4.1 所示。

属性名称 Font name 中的字体列表来自于操作系统自带的字体以及 Font Extension 字体扩展，这些都是受 JVM 管理的，这些列表中的字体根据不同的操作系统而不同，所以如果想使用操作系统自带的字体，请在不同的操作系统中安装这些字体，更好的办法是创建一个 Font Extension 字体扩展，因为每种操作系统所带的字体名称都不一样，使用字体扩展后可以将*.ttf 文件转成*.jar 文件，只要软件项目拥有这个.jar 文件就可以在不同的操作系统中进行使用。

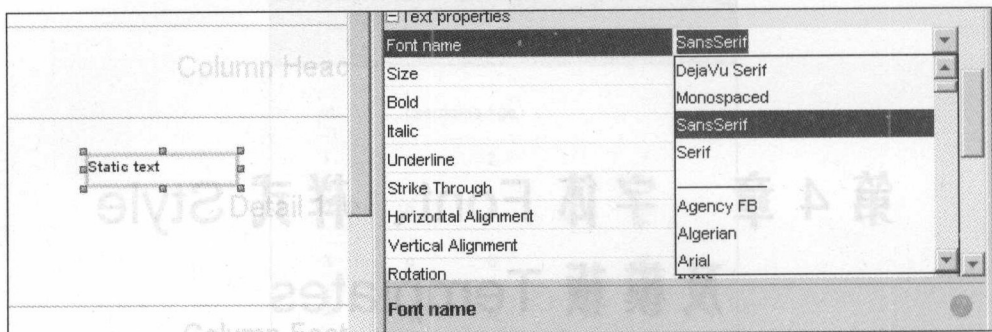


图 4.1 设置字体常用属性

在导出 PDF 文件时，图 4.1 中的 Font name 属性就无效了，必须使用如图 4.2 所示的属性设置 PDF 字体。

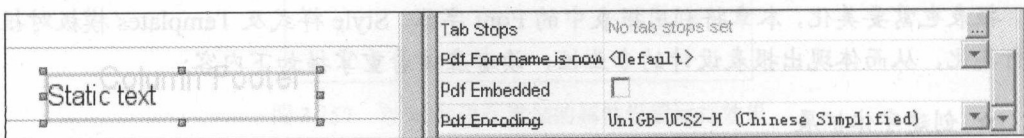


图 4.2 导出 PDF 文件时需要设置字体的属性

4.1.1 使用自带字体

PDF 文件可以支持的字体类型如图 4.3 所示。

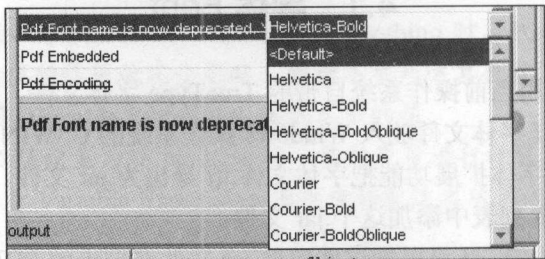


图 4.3 PDF 文件支持的字体

图 4.3 中的列表是 PDF 文件支持的全部字体，可以做以下实验：在报表模板中添加两个 Static Text 控件，设置第 1 个控件的属性如图 4.4 所示。

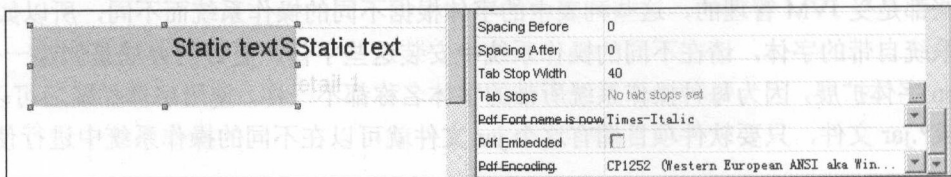


图 4.4 选择 PDF 文件自带的粗体字体

设置第 2 个 Static Text 控件属性如图 4.5 所示。

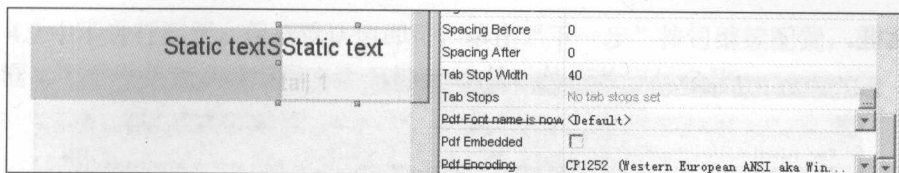


图 4.5 选择 PDF 文件默认字体

程序运行后的效果如图 4.6 所示。

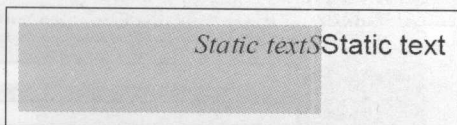


图 4.6 使用 PDF 自带字体

这就说明 PDF 文件在默认情况下是支持这些字体的。

4.1.2 使用第三方字体

前面已经介绍过如何使用 PDF 默认的字体系式，那如果想在 PDF 文件中使用第三方的字体，该如何配置呢？

第三方字体只有.ttf 文件，所以要把.ttf 文件导入到 iReport 软件中，单击图 4.7 中的 **Install Font** 按钮，进行.ttf 字体的安装。

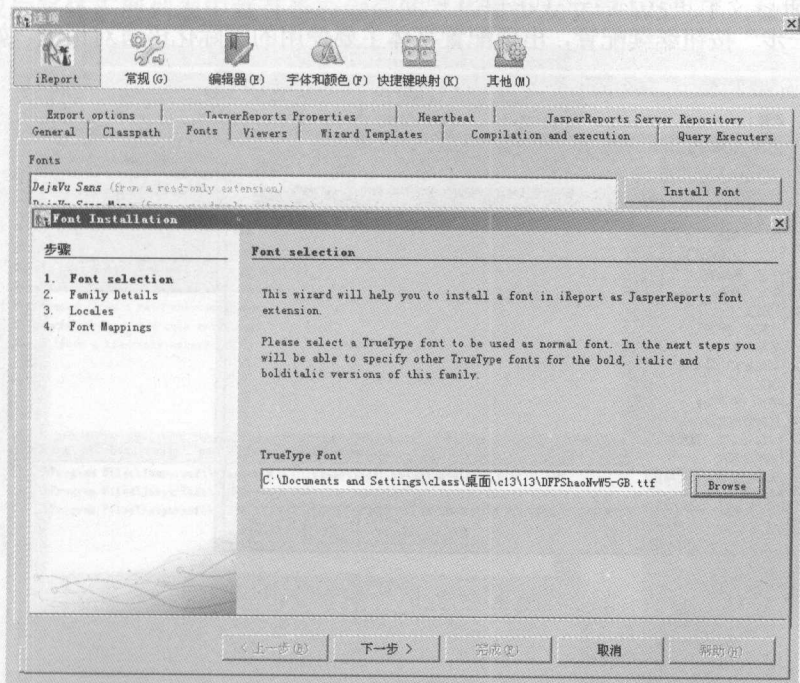


图 4.7 安装.ttf 字体

选择一个字体文件 DFPShaoNvW5-GB.ttf，单击“下一步”按钮继续配置，出现如图 4.8

所示的对话框。

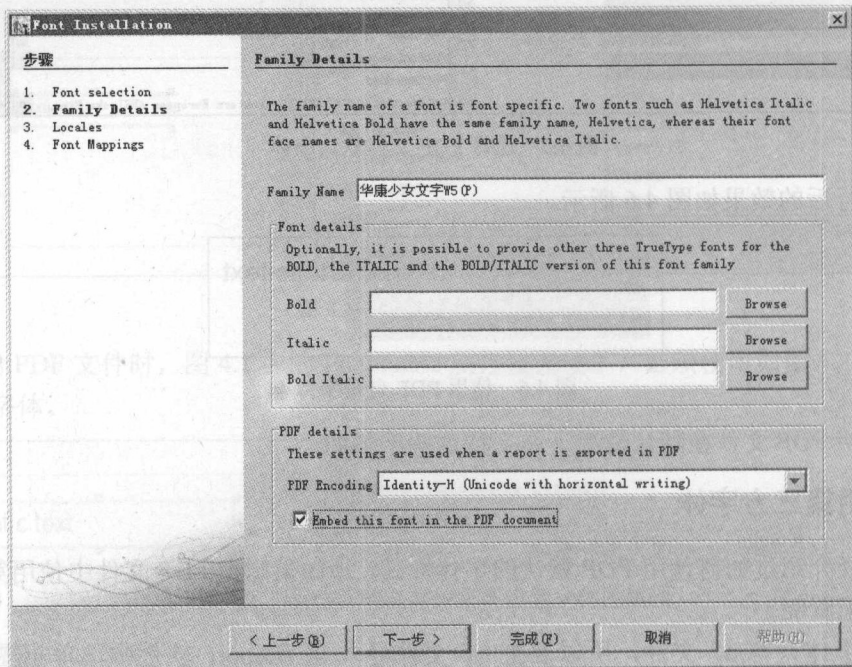


图 4.8 配置字体编码及是否嵌入到 PDF 文件中

单击“下一步”按钮继续配置，出现配置字体主要使用的国际化语言对话框，如图 4.9 所示。

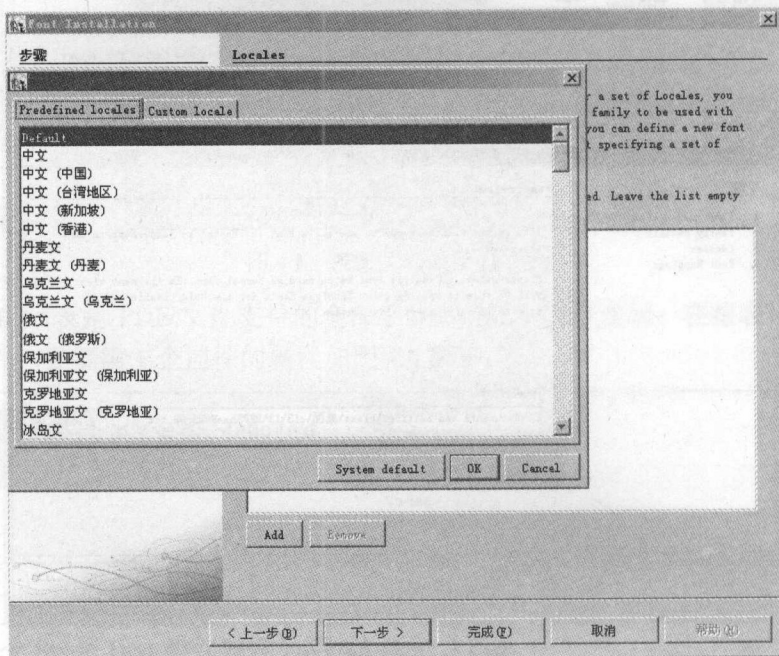


图 4.9 配置字体主要使用的国际化语言

在图 4.9 中不进行配置，保持默认值即可，单击“下一步”按钮继续配置，出现如图 4.10 所示对话框，用于设置当把报表导出为 HTML 文件时使用的字体名称。

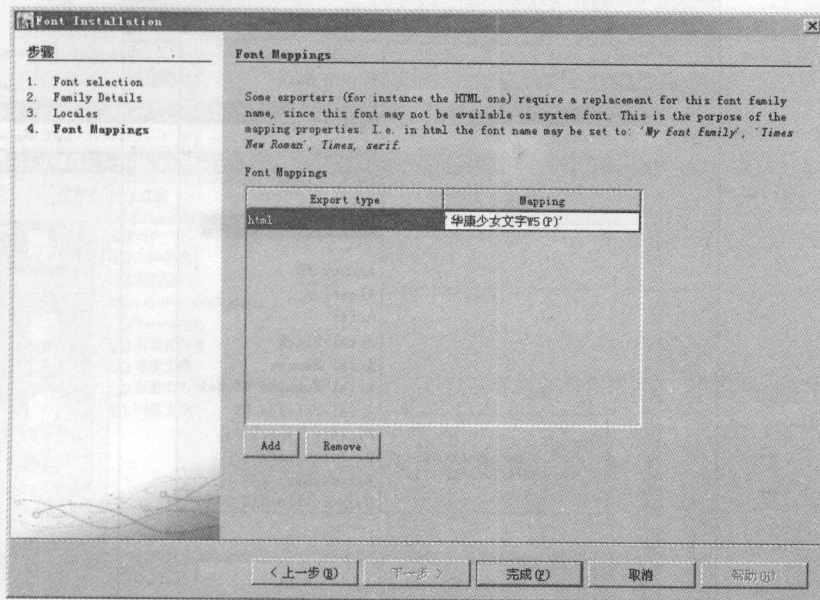


图 4.10 设置字体名称

配置完成后字体扩展列表中就有当前创建的“华康少女文字 W5”了，如图 4.11 所示。

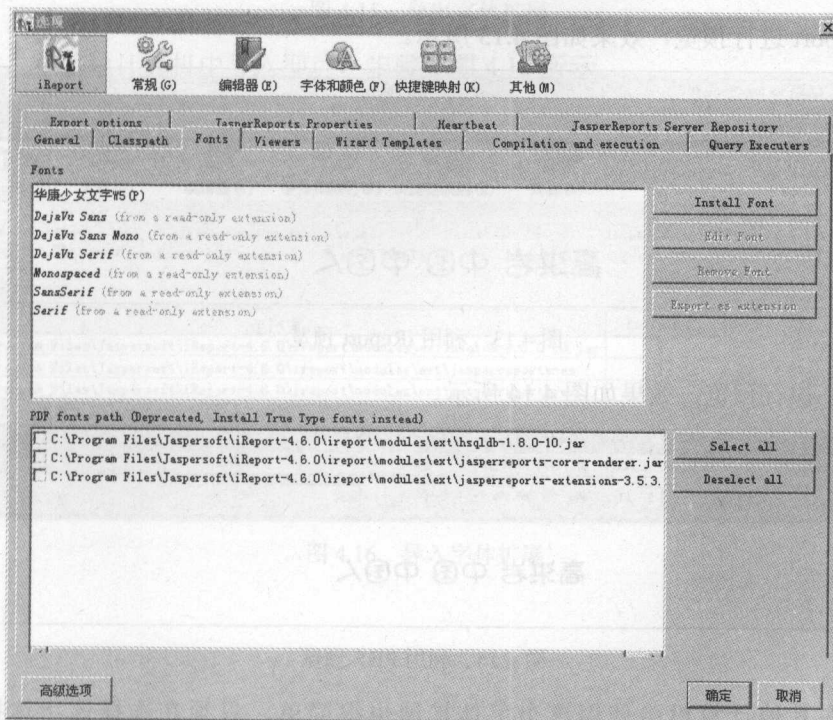


图 4.11 出现华康字体扩展

新建一个报表模板，设置 Static Text 控件的字体如图 4.12 所示。



图 4.12 选择“华康少女文字 W5”

利用 iReport 进行预览，效果如图 4.13 所示。

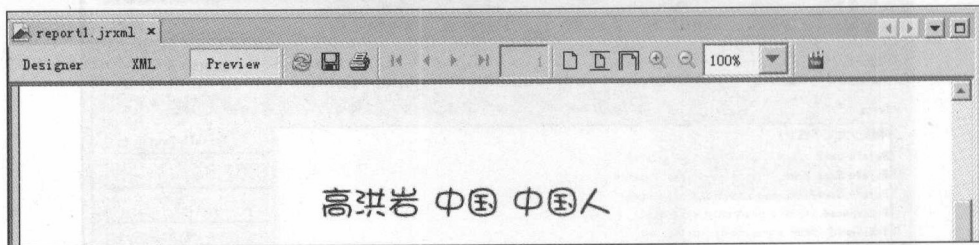


图 4.13 利用 iReport 预览

利用 PDF 进行预览，效果如图 4.14 所示。

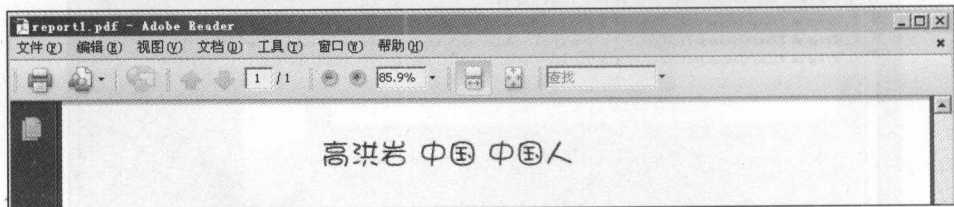


图 4.14 利用 PDF 预览

如果想在其他计算机中使用这个字体扩展也很简单，只须在选择该字体扩展后单击

Export as extension

按钮，在弹出的对话框中保存为.jar 文件即可，如图 4.15 所示。

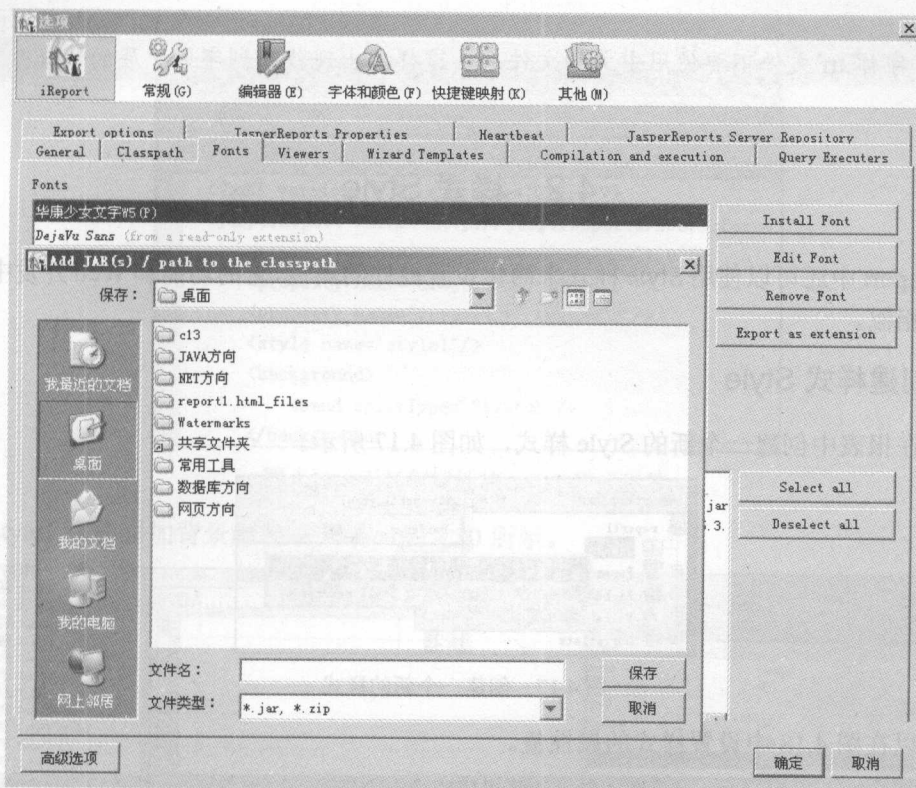


图 4.15 导出字体扩展

应用时，在其他计算机中导入即可，步骤如图 4.16 所示。

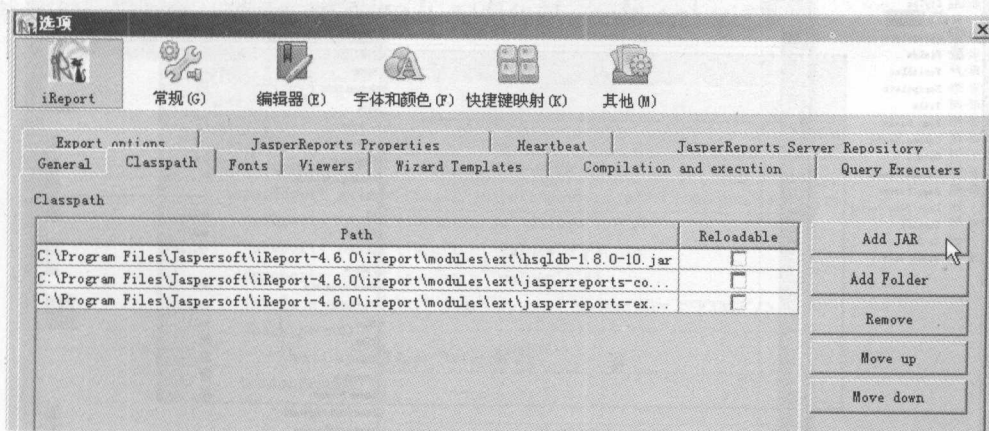


图 4.16 导入字体扩展



提示

字体.ttf 文件不要使用中文的文件名，这样会出现找不到字体扩展的情况。

4.2 样式 Style

在 iReport 中还可以使用 Style 样式来简化设置控件的属性,它的功能和 Web 开发中的 CSS 样式功能相似。

4.2.1 创建样式 Style

可以在报表中创建一个新的 Style 样式,如图 4.17 所示。

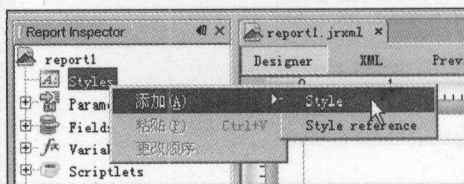


图 4.17 创建一个新的样式

还可以在图 4.18 中设置样式的属性值。

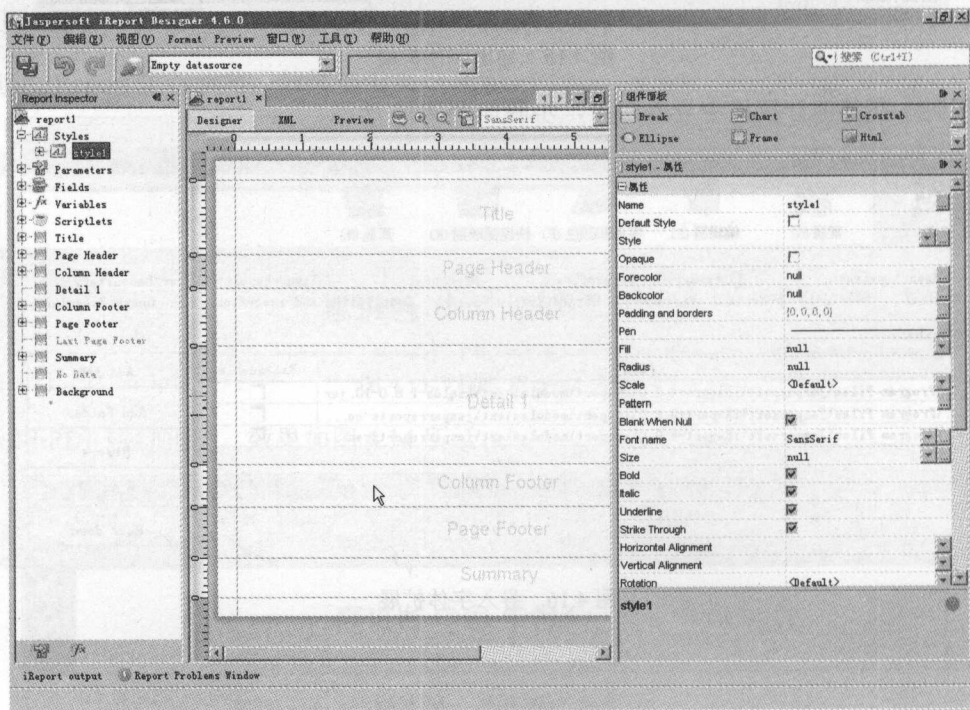


图 4.18 设置 style1 样式的属性值

新建一个样式 Style 后,在 .jrxml 文件中添加一个无任何属性值的 <style> 标签,效果如图

4.19 所示。

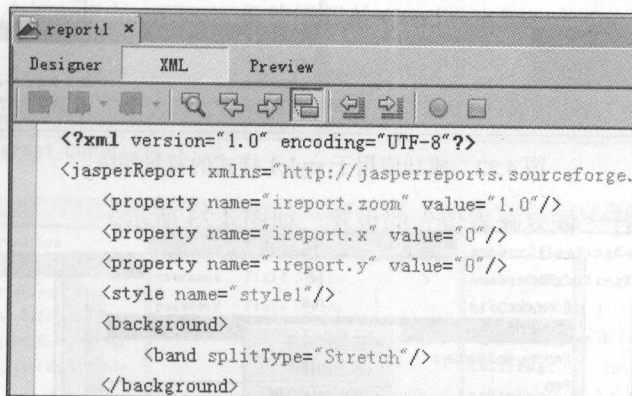


图 4.19 无任何属性的 style1 样式标签

对 style1 样式添加背景颜色，效果如图 4.20 所示。

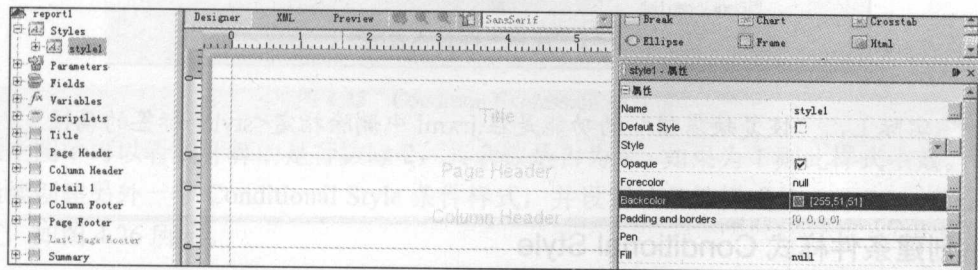


图 4.20 添加背景颜色样式

然后在.jrxml 文件中添加对应的 XML 配置代码，如图 4.21 所示。

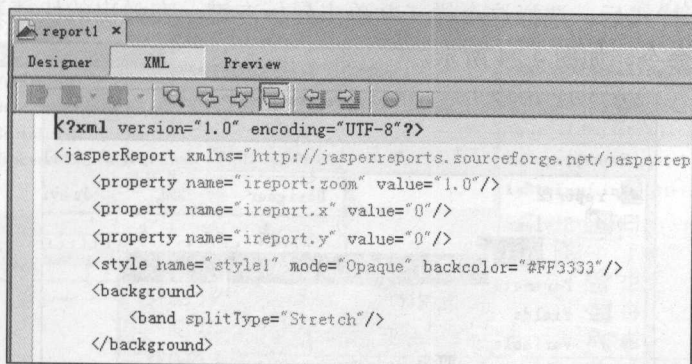


图 4.21 在.jrxml 文件中添加代码

拖曳一个 Static Text 控件到报表模板并应用样式，效果如图 4.22 所示。

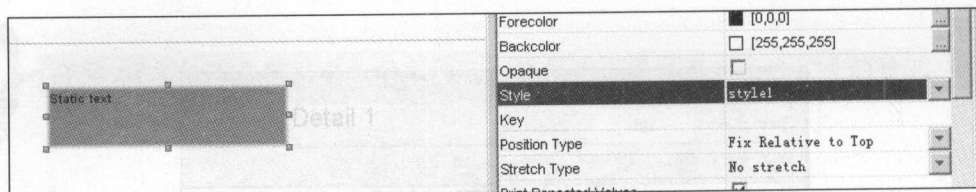


图 4.22 成功应用于 style1 样式的背景颜色

还可以对样式进行“恢复缺省值”的设置，如图 4.23 所示。

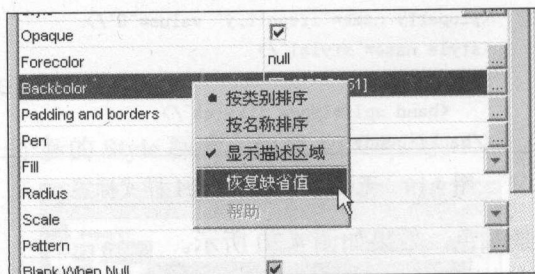


图 4.23 选择“恢复缺省值”



实际上，“恢复缺省值”的功能是在.jrxml 中删除指定<style>标签的属性。

提示

4.2.2 创建条件样式 Conditional Style

条件样式 Conditional Style 来自于一个存在的样式，但笔者建议这个存在的样式无任何被更改过的属性，所以建议新建一个报表以及一个 Style 样式。

创建完 style1 样式后，选择这个样式并单击鼠标右键，在弹出的快捷菜单中选择“添加 Conditional Style”命令，如图 4.24 所示。

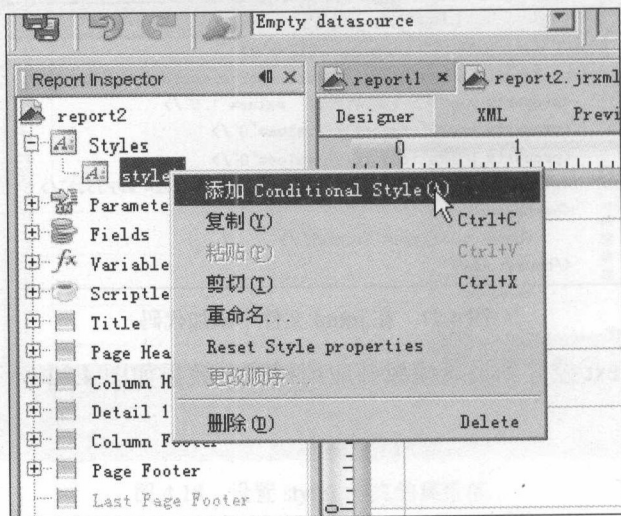


图 4.24 新建一个 Conditional Style 条件样式

创建完成后, 如果想在报表中使用 Conditional Style 条件样式, 则必须配置 Condition Expression 属性, 第 1 个 Conditional Style 条件样式的 Condition Expression 属性值设置如图 4.25 所示。

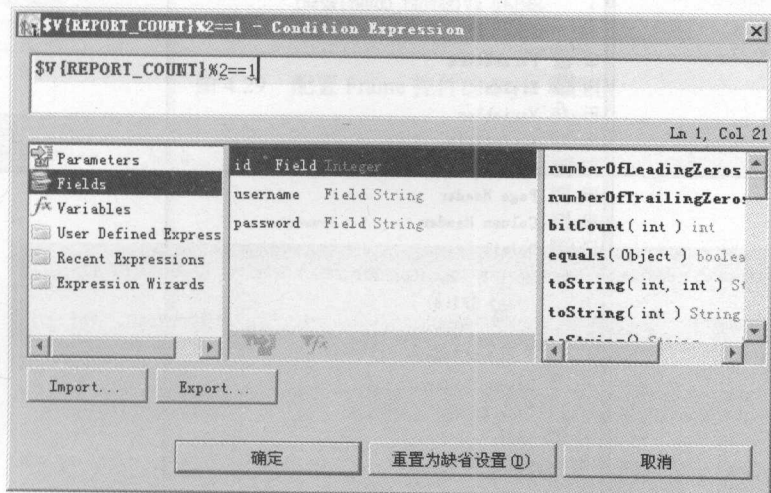


图 4.25 Condition Expression 属性值

从上图中可以看到计算的是行数除 2, 其余数是否为 1, 如果为 1 则此样式生效。

继续创建另外一个 Conditional Style 条件样式, 并设置该条件样式的 Condition Expression 属性值, 如图 4.26 所示。

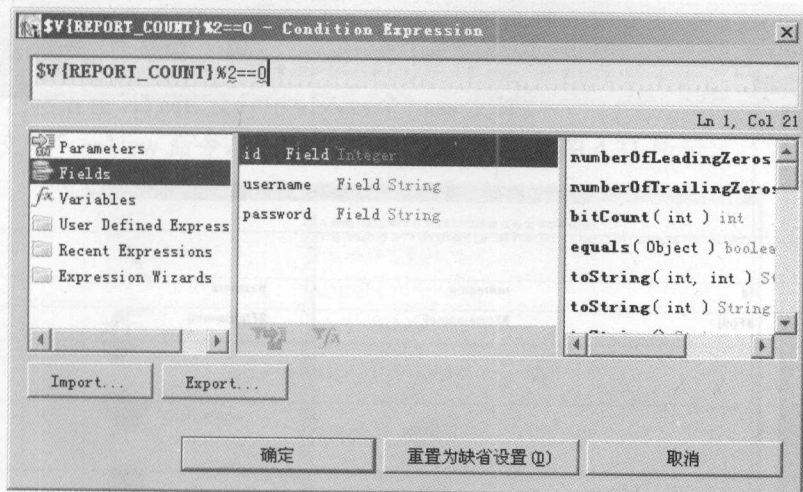


图 4.26 Condition Expression 属性值

Conditional Style 条件样式创建完毕后设计报表模板, 在 Detail 1 栏中添加 4 个控件, 如图 4.27 所示。

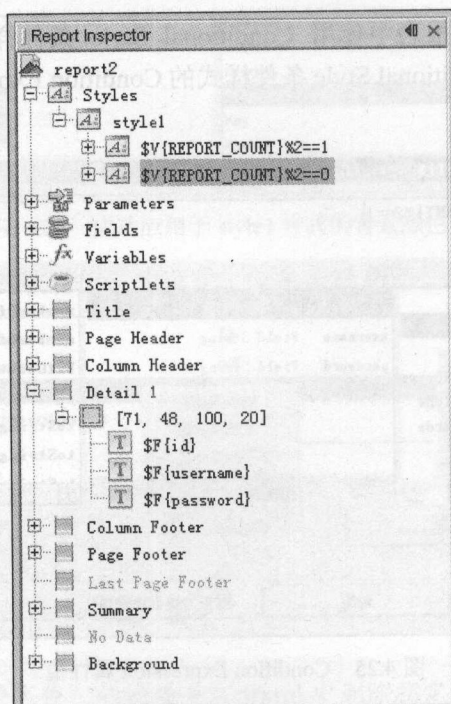


图 4.27 在 Detail 1 栏中添加 4 个控件

在 Detail 1 栏中的控件布局如图 4.28 所示。

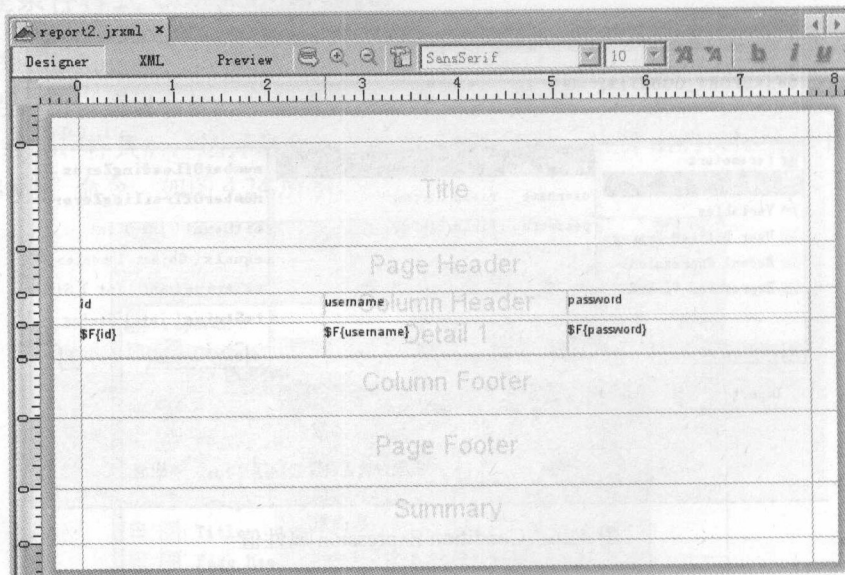


图 4.28 显示数据表 userinfo 中的数据记录

最为关键的是配置 Frame 控件的 Style 属性值，如图 4.29 所示。
运行效果如图 4.30 所示。

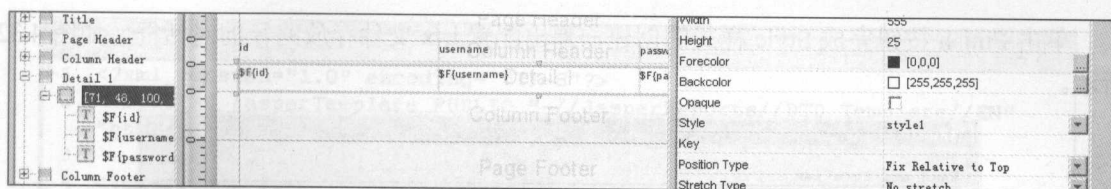


图 4.29 配置 Frame 控件的 Style 属性值

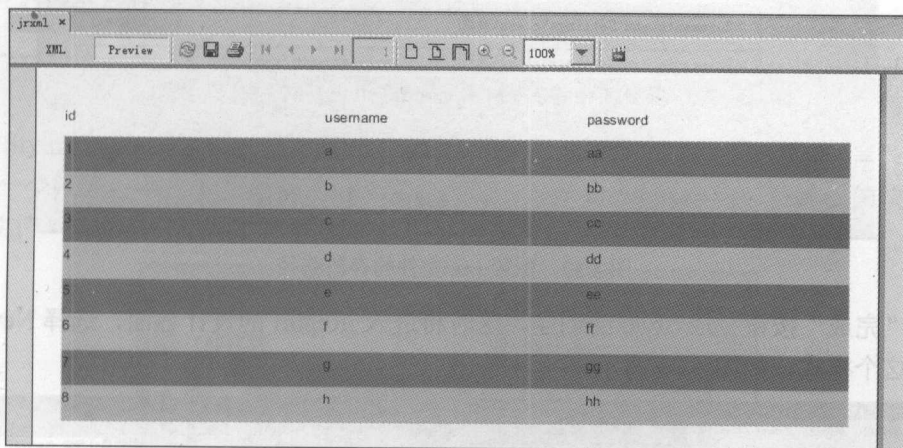


图 4.30 实现了隔行变色的效果

4.2.3 创建通用样式 Style

在 iReport 中还可以创建一个 .jrtx 文件，该文件的主要作用就是配置 Style 的集合，然后在不同的报表中可重用这个 .jrtx 文件，即重用 Style 样式，非常类似于在 Web 开发中将样式放入 *.css 文件中，然后在 HTML 文件中引入这个 *.css 文件。

选择“文件”→New 命令，弹出 New file 对话框，如图 4.31 所示。

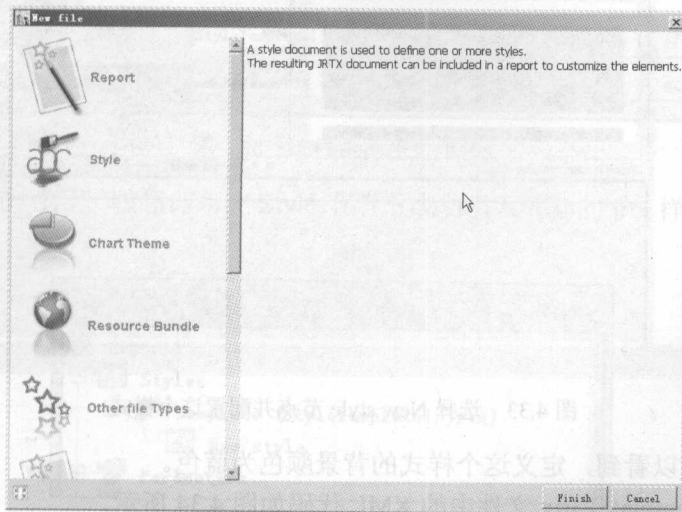


图 4.31 新建 Style 样式

单击 Finish 按钮完成 Style 样式的创建，弹出配置.jrtx 文件存放路径的对话框，如图 4.32 所示。

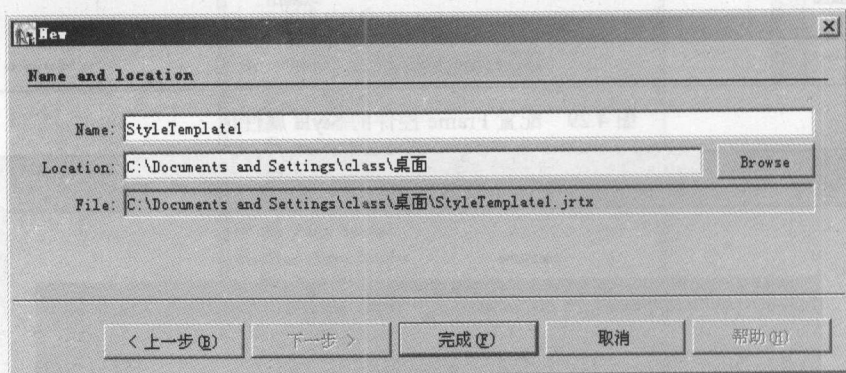


图 4.32 配置.jrtx 文件的存放路径

单击“完成”按钮完成 Style 的创建，这时将进入 iReport 的设计界面，选择 New style 节点并配置这个样式，如图 4.33 所示。

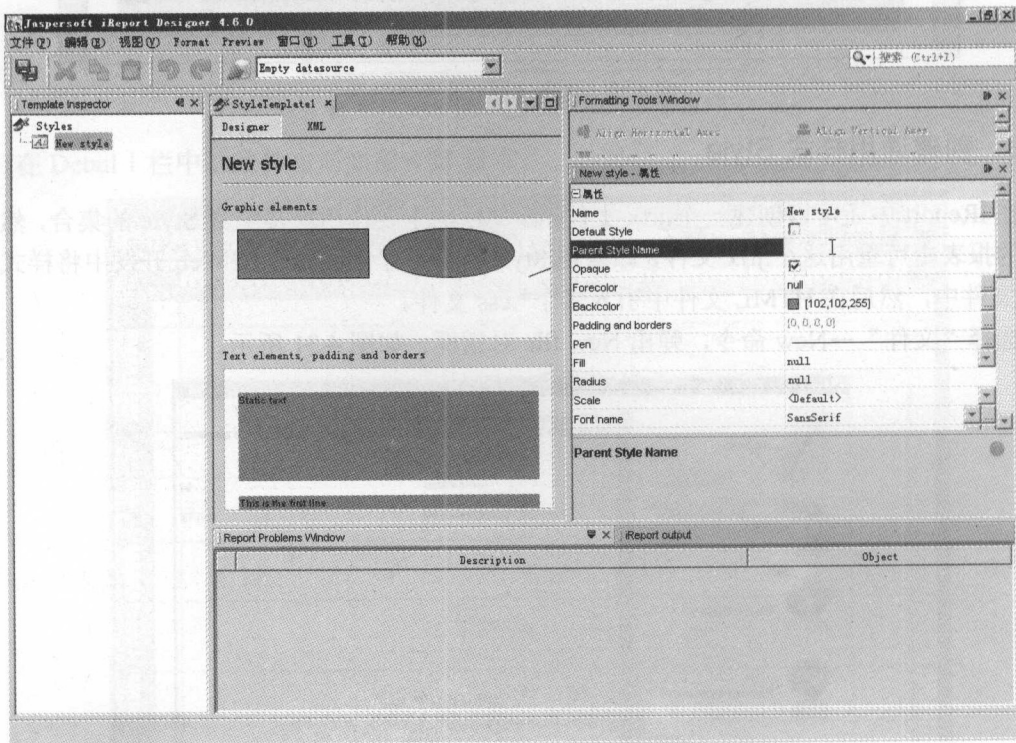


图 4.33 选择 New style 节点并配置这个样式

从图 4.33 中可以看到，定义这个样式的背景颜色为蓝色。生成的 StyleTemplatel.jrtx 文件中的 XML 代码如图 4.34 所示。


```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE jasperTemplate PUBLIC "-//JasperReports//DTD Template//EN"
  "http://jasperreports.sourceforge.net/dtds/jasptemplate.dtd">
3
4 <jasperTemplate>
5     <style name="New style" mode="Opaque" bgcolor="#6666FF"/>
6 </jasperTemplate>
7

```

图 4.34 StyleTemplate1.jrtx 文件中的代码

这个 StyleTemplate1.jrtx 文件中就是 Style 样式的配置。

新建一个报表，添加一个外部的.jrtx 文件，即右键单击 Style 选项，在弹出的快捷菜单中选择“添加”→Style reference 命令，如图 4.35 所示。

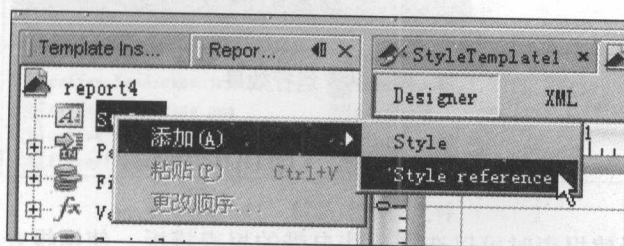


图 4.35 添加.jrtx 文件

选择.jrtx 所在的路径和文件名，如图 4.36 所示。

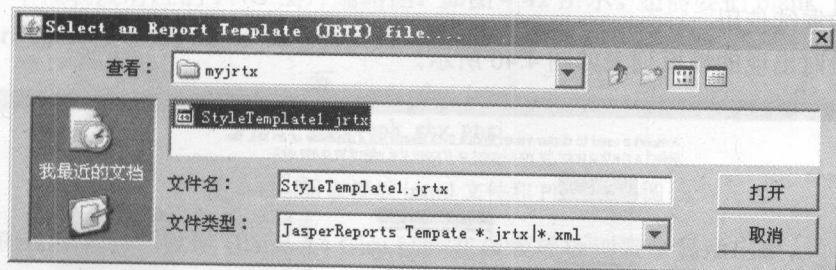


图 4.36 选择路径及文件名

选择.jrtx 后单击“打开”按钮即可在 Style 节点下成功引入外部的.jrtx 样式，效果如图 4.37 所示。

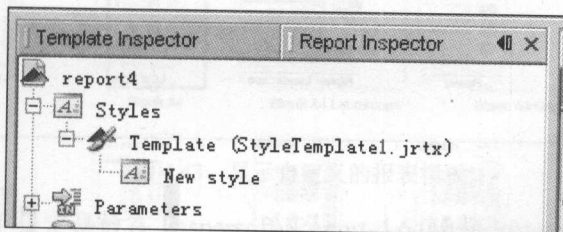


图 4.37 成功引入外部.jrtx 文件

添加一个 Static Text 控件，设置 Style 属性如图 4.38 所示。

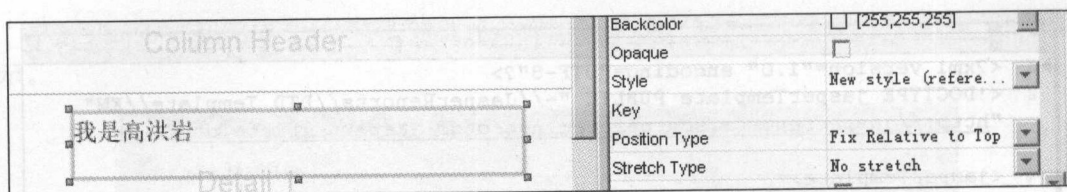


图 4.38 设置 Style 属性

运行效果如图 4.39 所示。

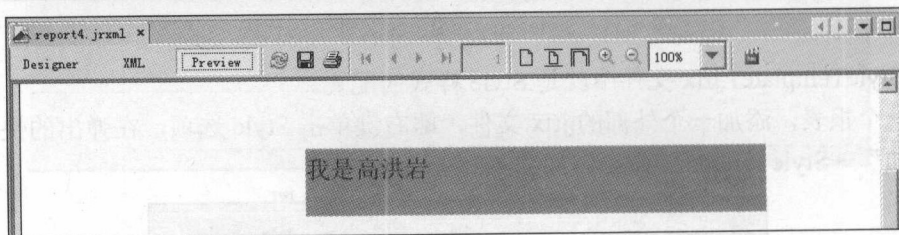


图 4.39 运行效果

4.3 模板 Templates

在使用 iReport 新建报表时可以选择一些自带的报表模板，使用报表模板可以大大提高开发的效率，在原有报表模板的基础上进行小范围的修改，而且模板中的样式，如字体大小、文字颜色、背景颜色等属性还可以得到充分的保留，把设计界面美观的报表保存为模板，以便在后面的开发中继续使用。

新建报表时出现的模板列表如图 4.40 所示。

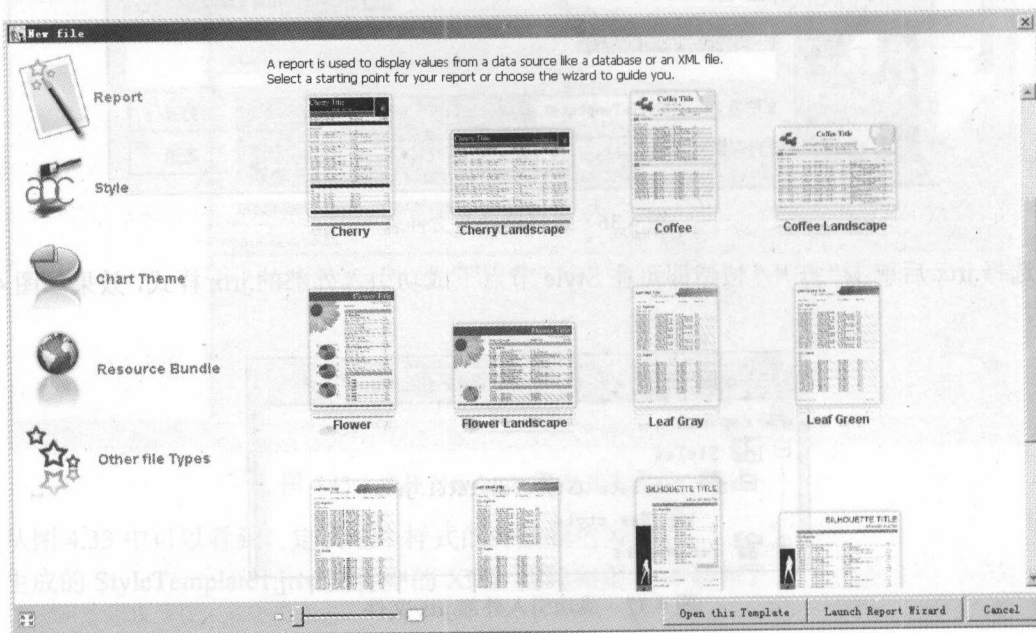


图 4.40 模板列表

这些默认的报表模板存放在 Jaspersoft\iReport-4.6.0\ireport\templates 路径中, 如图 4.41 所示。

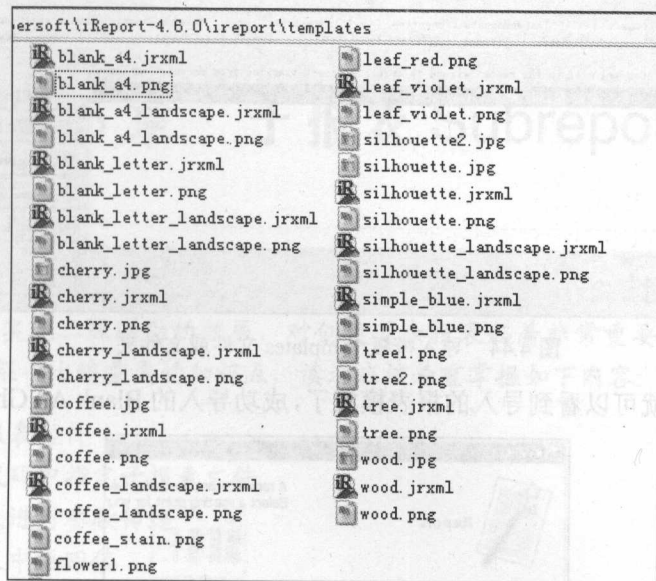


图 4.41 默认模板文件

从图 4.41 中可以看到, 所谓的报表模板其实就是 .jrxml 文件, 并且还有对应的 PNG 图片缩略图, 以显示报表的整体外观, 在 iReport 启动时就可以加载这些报表模板。

新建一个报表及对应的 PNG 图片缩略图, 如图 4.42 所示。还需要把 blank_ghy.jrxml 文件中的 <jasperReport> 标签的 name 属性值改为 “Blank A4 Ghy”。

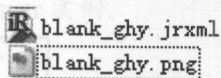


图 4.42 报表模板 .jrxml 文件和 PNG 缩略图

重启 iReport 软件, 发现 Blank A4 Ghy 已经在报表模板列表中显示出来, 如图 4.43 所示。

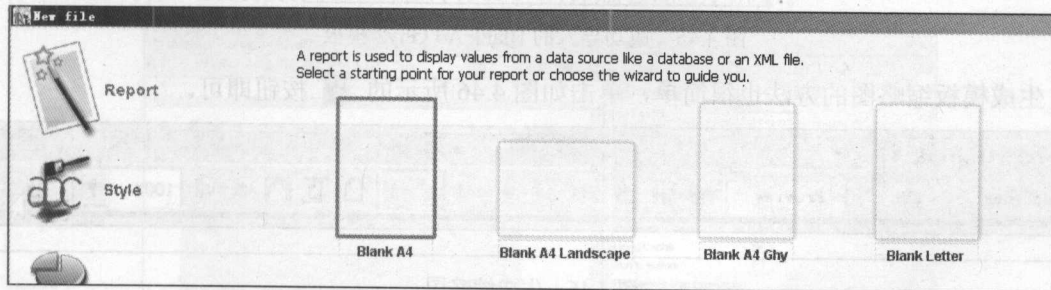


图 4.43 显示自定义的报表模板

当然, 报表模板不一定是放在 Jaspersoft\iReport-4.6.0\ireport\templates 路径中, 还可以放在其他路径, 只须使用如图 4.44 所示的导入模板 Templates 文件或文件夹的功能进行导入即可。

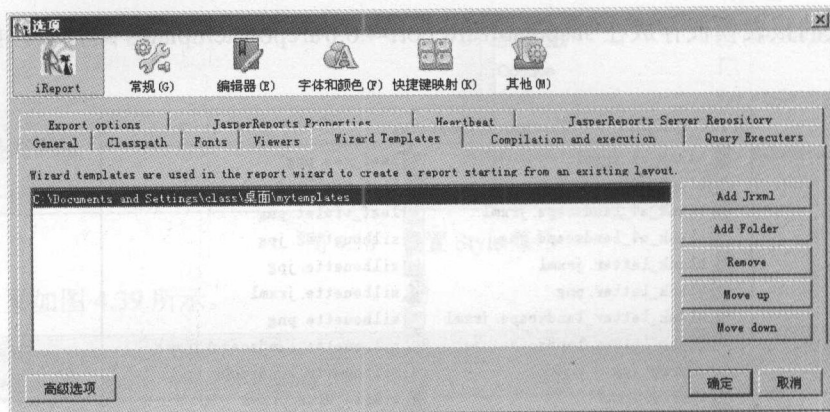


图 4.44 导入模板 Templates 文件或文件夹

重启 iReport 后就可以看到导入的报表模板了,成功导入的 Blank A4 Ghy2 如图 4.45 所示。

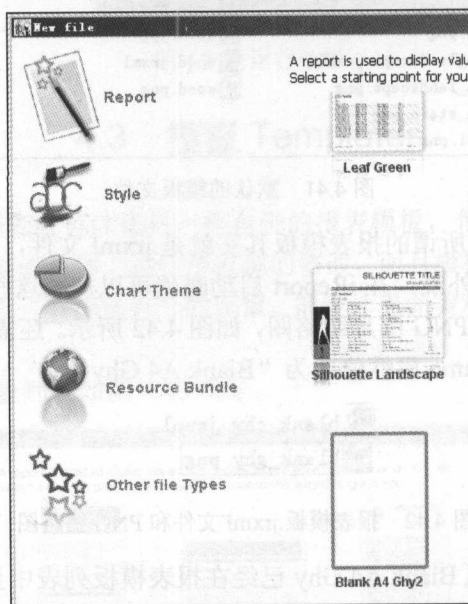


图 4.45 成功导入的 Blank A4 Ghy2 模板

生成模板缩略图的方法也很简单,单击如图 4.46 所示的  按钮即可。

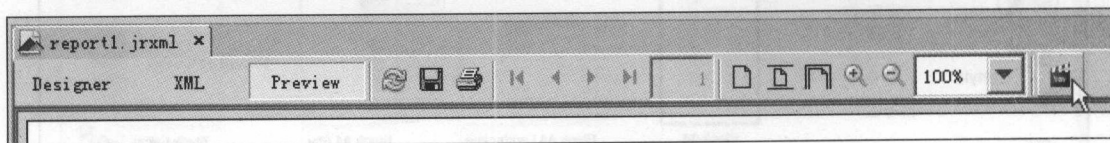


图 4.46 生成缩略图

第 5 章 子报表 Subreport

↓ 导言

子报表可以实现二级联动的效果，对创建复杂报表起着非常重要的作用。子报表是 JasperReports 框架中比较重要的知识点，读者应该着重掌握如下内容：

- ★ 使用子报表控件
- ★ 在程序代码中指定子报表文件
- ★ 对子报表进行参数传递
- ★ 从子报表中返回值

5.1 子报表 Subreport 的基础知识

其实子报表的解释很简单，就是报表中有报表。

如果想在 iReport 中使用子报表 Subreport 必须要考虑 3 件事情：

- 子报表如何取得对应的 .jasper 实例。
- 如何将数据传递给子报表。
- 如何给子报表参数设置传递数据值。

在解决这 3 个问题之前可以尝试着先把 Subreport 控件放到报表模板中，然后查看一下该控件独有的属性，Subreport 控件的属性及控件的外观如图 5.1 所示。

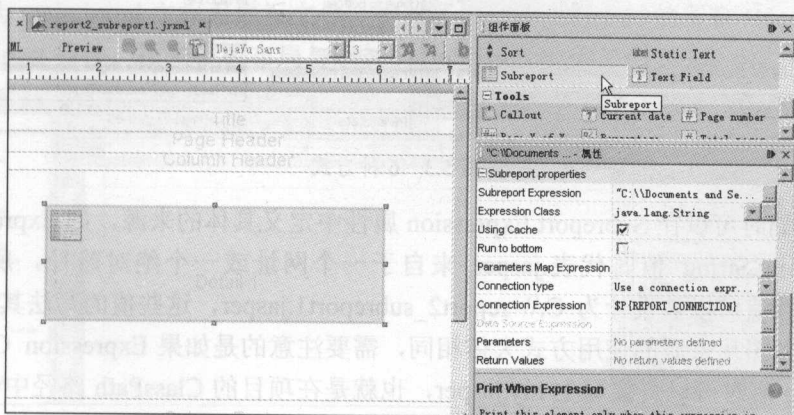


图 5.1 Subreport 控件的属性及控件的外观

想要解决上面这 3 个问题，也就是正确地使用 Subreport 子报表，其实答案就在图 5.1 中的属性里，在后面的章节中将主要介绍属性的作用及功能演示，这里不再赘述。

5.1.1 子报表 Subreport 的.jasper 文件来源

子报表 Subreport 其实就是一个 .jrxml 文件，更具体来讲，使用子报表其实就是在使用 .jasper 文件，这也代表如果在 a.jrxml 中使用 Subreport 的功能，那么必须还要有对应的子报表文件 a_subreport.jrxml，也就是使用两个 .jrxml 文件来实现这个主报表中有子报表的功能，即两个 .jasper 文件具有彼此可以互相调用、互相传递数据的功能。

将 Subreport 控件拖曳到 Detail 1 栏中，弹出一个配置对话框，如图 5.2 所示。

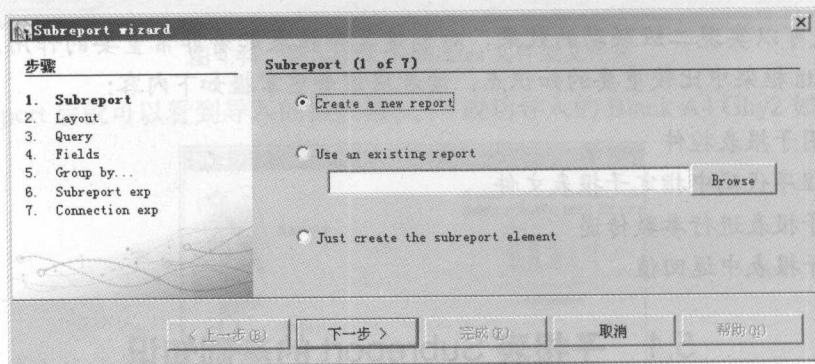


图 5.2 配置子报表.jrxml 来源

在图 5.2 中可以设置是新建一个子报表还是使用存在的 .jrxml 文件当做子报表的 .jrxml 文件来源，这样配置是使用指定路径中的指定 .jrxml 文件名来定义子报表的 .jrxml 文件来源，也就是使用绝对路径。还可以在 Subreport 控件的 Expression Class 中定义 Subreport 子报表关联的 .jasper 来源，一共有 6 种方式，如图 5.3 所示。

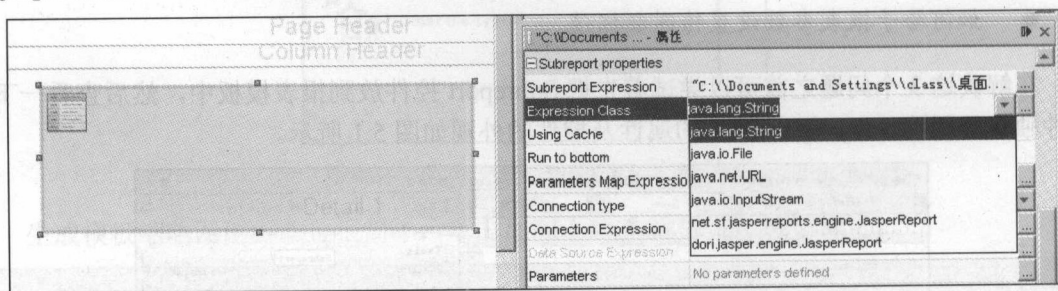


图 5.3 6 种方式

在使用它们时可以在 Subreport Expression 属性中定义具体的来源，如 Expression Class 属性选择 java.lang.String 值时代表 .jasper 来自于一个网址或一个绝对路径，所以 Subreport Expression 属性值可以被设计为 C:\\ report2_subreport1.jasper，这些值的写法其实和前面使用 Image 控件定义图片来源的使用方式大体相同，需要注意的是如果 Expression Class 属性选择 java.lang.String 值时可以直接写上 my.jasper，也就是在项目的 ClassPath 路径中寻找 my.jasper 文件，并不需要写上完整的路径，这种情况非常适合于以 Web 为基础的 JavaEE 应用程序中，

这个知识点已经在第 1 章中做过介绍, 这里不再赘述。

另外还可以在 Servlet 中利用传递路径参数 Parameters 的形式来动态指定.jasper 文件所在的路径, 当然如果这样做, 就要多传递一个 Paramaters 参数对象作为.jasper 文件的存放路径, 使用下面的方式配置 Subreport Expression 属性值就可以实现 Parameters 变量对象传递路径的功能:

```
$P{SUBREPORT_DIRECTORY} + "ghy.jasper"
```

使用 Subreport 子报表创建向导时可以自动生成这个参数, 自动生成的子报表所在路径的 Parameters 参数如图 5.4 所示。

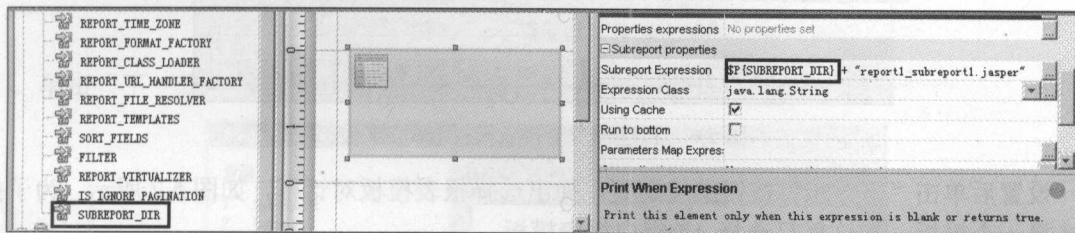


图 5.4 自动生成的子报表所在路径的参数

通过上面的操作, 就可以在主报表中使用子报表所关联的.jasper 文件了。

有了子报表 Subreport, 但子报表中并没有数据, 可以通过两种方式来填充子报表中的数据, 一种是 JDBC, 另外一种 DataSource 数据源, 使用 DataSource 数据源的方式填充子报表可以有多种数据源类型, 关于不同 DataSource 数据源的用法将在后面的章节中介绍, 这里不再赘述, 本节主要是利用 JDBC 数据源填充子报表。

5.1.2 子报表 Subreport 的示例——静态文本

本示例将演示主、子报表的使用, 并且主、子报表中的数据都是静态文本, 学习这个实验的主要目的是熟悉 Subreport 在 iReport 软件中的使用, 需要注意的是: 如果使用主、子报表时, 子报表的 DataSource 中没有数据, 则在预览主报表时子报表默认是不显示的。

1. 创建主报表及子报表

新建名称为 main.jrxml 主报表模板文件, 把除了 Detail 1 栏之外的所有 Band 删除, 并且添加一个 Subreport 控件, 报表模板布局与添加的 Subreport 控件如图 5.5 所示。

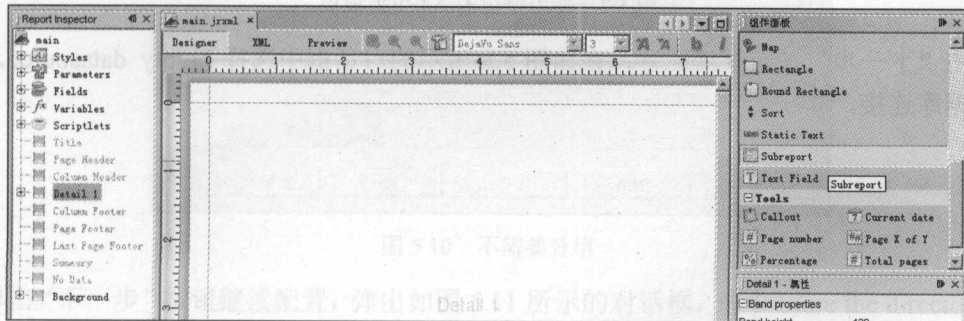


图 5.5 报表模板布局与添加 Subreport 控件

弹出询问是否新建一个 .jrxml 文件还是使用存在的 .jrxml，在这里选择新建一个 .jrxml，如图 5.6 所示。

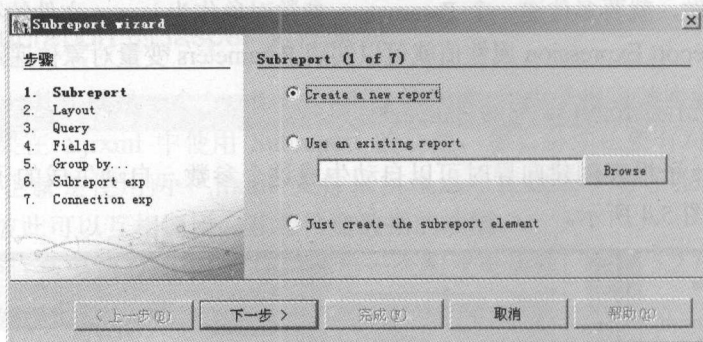


图 5.6 新建一个子报表模板 .jrxml 文件

设置后单击“下一步”按钮继续配置，弹出选择报表模板对话框，如图 5.7 所示，为子报表选择一个模板，在这里只选择 A4 空白报表模板。

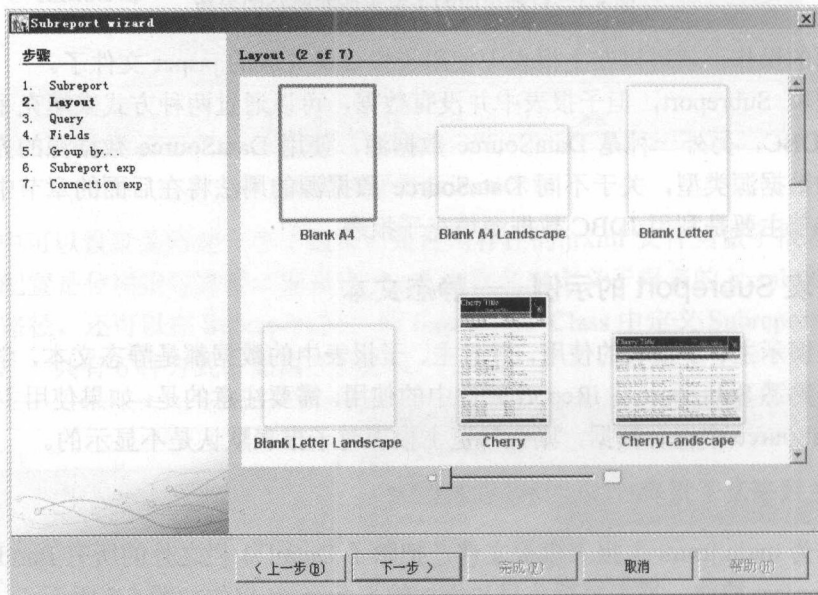


图 5.7 使用空白的 A4 报表模板

单击“下一步”按钮继续配置，在如图 5.8 所示的对话框中选择 Empty datasource，即不需要数据库连接。

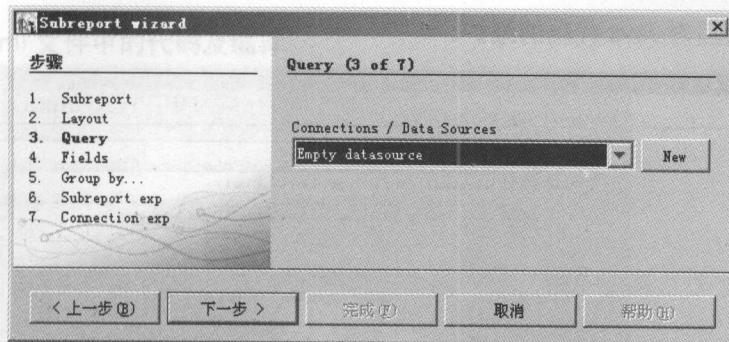


图 5.8 不需要数据库连接

单击“下一步”按钮继续配置，不选择任何 Field 对象，如图 5.9 所示。

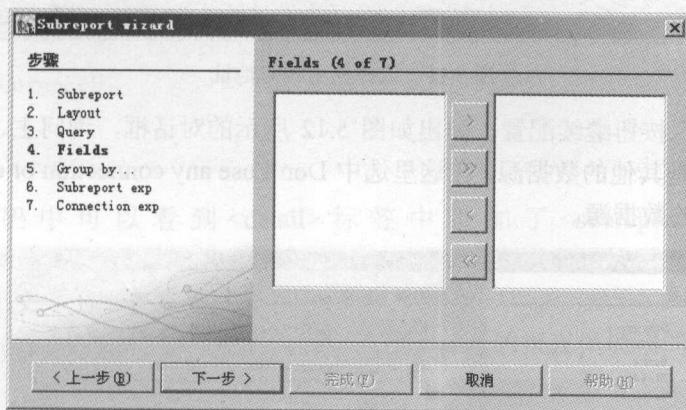


图 5.9 不选择任何 Field 对象

单击“下一步”按钮继续配置，不需要对子报表中的数据进行分组，如图 5.10 所示。

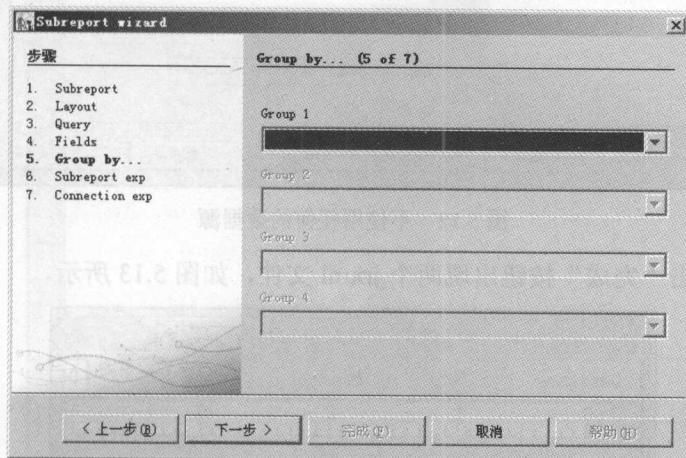


图 5.10 不需要分组

单击“下一步”按钮继续配置，弹出如图 5.11 所示的对话框，选中 Store the directory name in a parameter 单选按钮，也就是报表的路径可以来自于一个 Parameters 参数对象，这样可使

于.jasper 路径改变后对 Java 代码的维护。

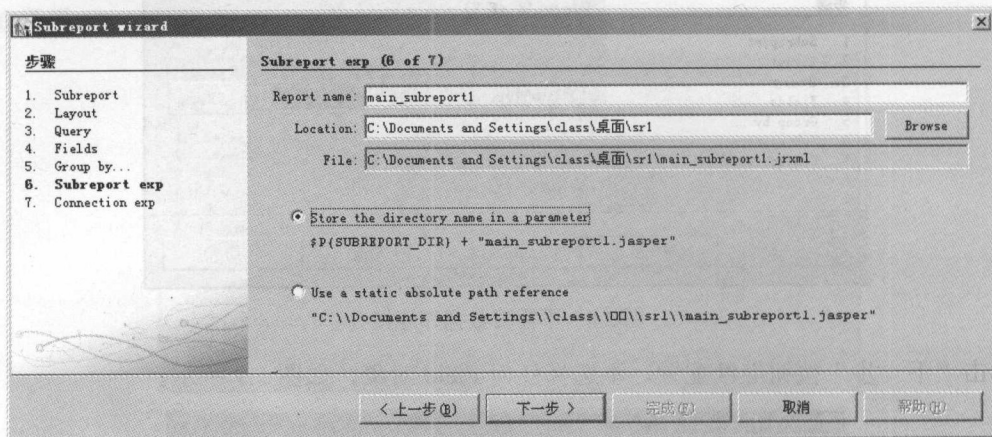


图 5.11 选择使用相对地址

单击“下一步”按钮继续配置，弹出如图 5.12 所示的对话框，询问主、子报表是使用同一个 Connection 还是其他的数据源，在这里选中 Don't use any connection or date source 单选按钮，即不使用任何的数据源。

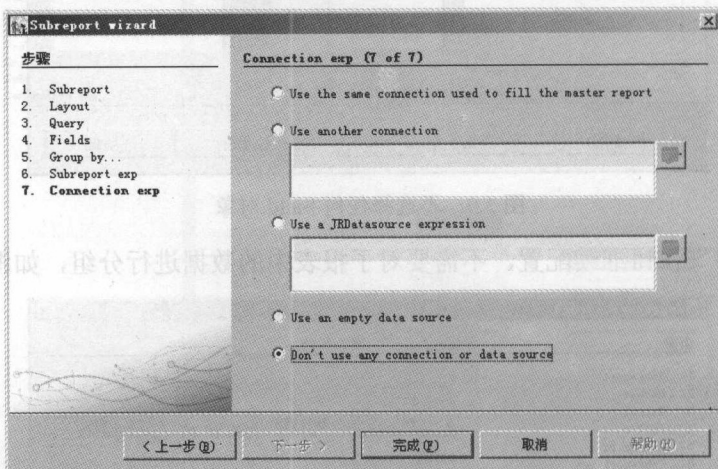


图 5.12 不使用任何的数据源

配置完成后单击“完成”按钮出现两个.jrxml 文件，如图 5.13 所示。

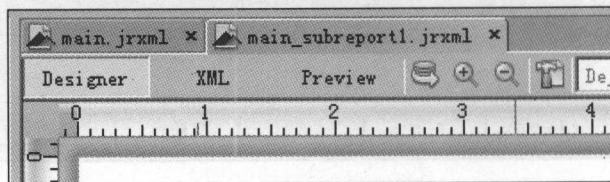


图 5.13 两个.jrxml 文件

2. 两个.jrxml 文件中的代码及编译

创建的 main.jrxml 文件中的核心代码如下：

```
<detail>
  <band height="264" splitType="Stretch">
    <subreport>
      <reportElement uuid="cc49377b-aba3-4bab-85fe-44e469c547fe"
        x="70" y="56" width="200" height="100" />
      <connectionExpression><![CDATA[${REPORT_CONNECTION}]]>
      </connectionExpression>
      <subreportExpression>
        <![CDATA[${SUBREPORT_DIR} + "main_subreport1.jasper"]]>
      </subreportExpression>
    </subreport>
  </band>
</detail>
```

从以上代码中可以看到 <detail> 标签中添加了 <subreport> 标签，并且用 <subreportExpression> 标签设置子报表对应的 .jasper 文件存放位置，使用 \${SUBREPORT_DIR} 参数来设置 .jasper 文件的所在路径。

子报表文件并没有生成什么特别的代码，外观和普通的报表模板相同，如图 5.14 所示。

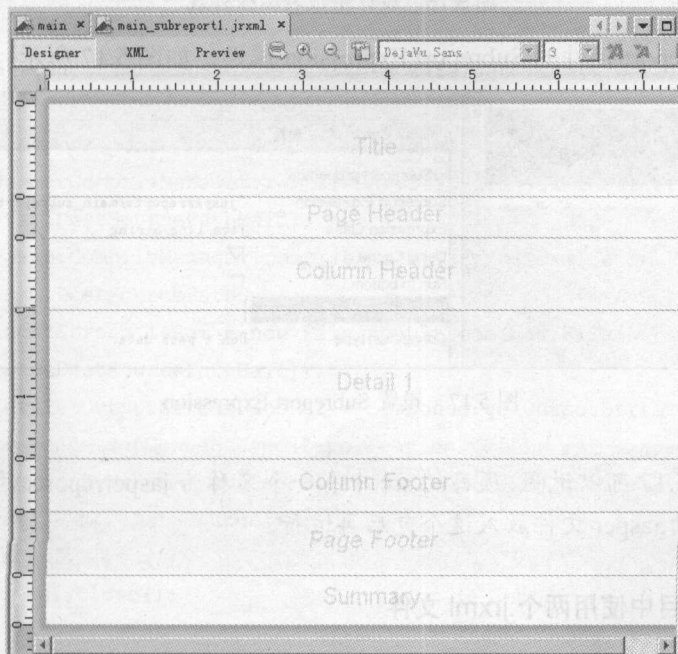


图 5.14 子报表使用的.jrxml 外观

在 main_subreport1.jrxml 文件中添加一个 Static Text 对象，设计报表模板样式如图 5.15 所示。

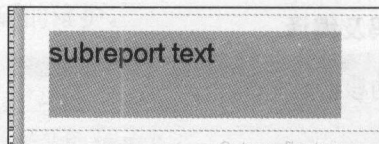


图 5.15 添加一个 Static Text 控件

设计 main.jrxml 文件模板样式，如图 5.16 所示。

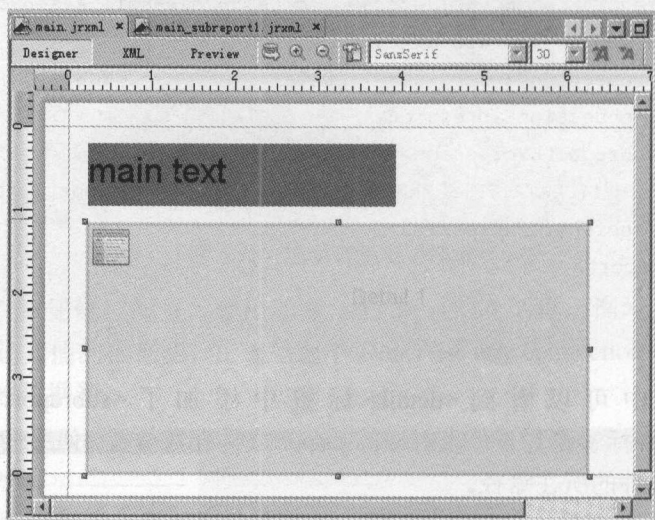


图 5.16 设计主报表模板外观

继续配置 Subreport 控件的 Subreport Expression 属性，如图 5.17 所示。

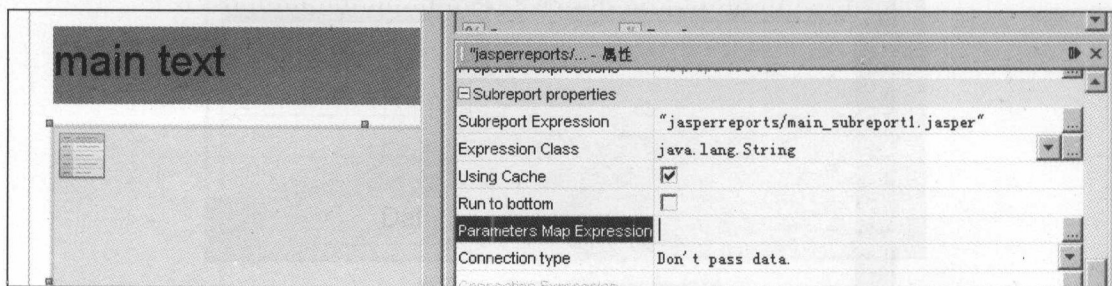


图 5.17 设置 Subreport Expression



通过图 5.17 可以说明，项目的 src 中有一个名称为 jasperreports 的包名，要把 jrxml 编译后的 .jasper 文件放入这个包后再运行。

提示

3. 在 Web 项目中使用两个 jrxml 文件

创建 Web 项目，将这两个 jrxml 文件放在 WebRoot 中的 report 文件夹下，项目结构如图 5.18 所示。

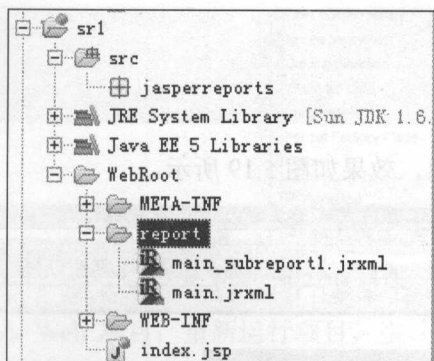


图 5.18 两个.jrxml 放入 report 文件夹下

创建 Servlet，核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
                + "report\\main.jrxml";
            String jrxmlSourcePathSub = this.getServletContext().getRealPath("/")
                + "report\\main_subreport1.jrxml";
            String jrxmlDestSourcePathMain = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1) + "jasperreports/main.jasper";
            String jrxmlDestSourcePathSub = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1) +
                "jasperreports/main_subreport1.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
                jrxmlDestSourcePathMain);
            JasperCompileManager.compileReportToFile(jrxmlSourcePathSub,
                jrxmlDestSourcePathSub);
            InputStream isRef = new FileInputStream(new File(
                jrxmlDestSourcePathMain));
            ServletOutputStream sosRef = response.getOutputStream();
            response.setContentType("application/pdf");
            JasperRunManager.runReportToPdfStream(isRef, sosRef, new HashMap(),
                new JREmptyDataSource());
            sosRef.flush();
            sosRef.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

运行后子报表并没有输出，效果如图 5.19 所示。

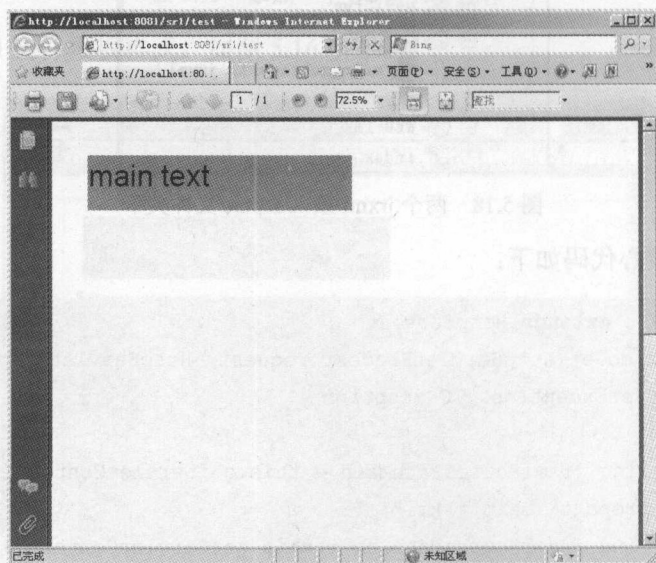


图 5.19 子报表并没有输出

因为子报表从未使用 DataSource，所以子报表的 Detail 1 栏不显示，这种情况该怎么处理呢？这时可以使用 No Data Section。

重新设计子报表模板，添加 “no data show!”，如图 5.20 所示。

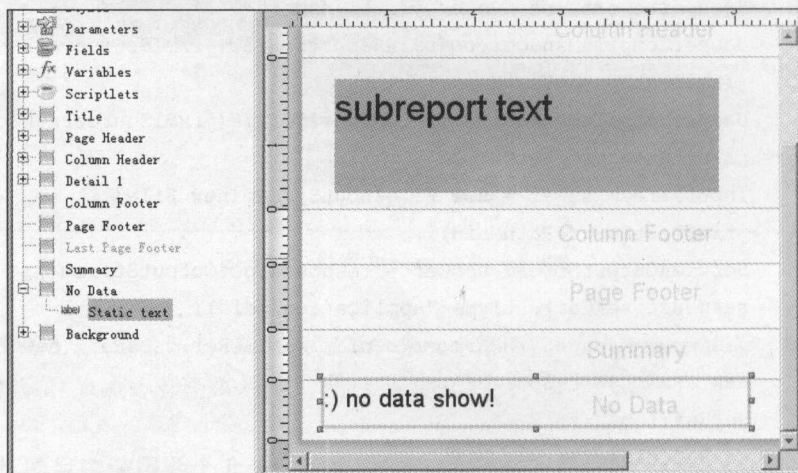


图 5.20 添加 “no data show!”

还需要设置子报表模板的 When No Data 属性，如图 5.21 所示。

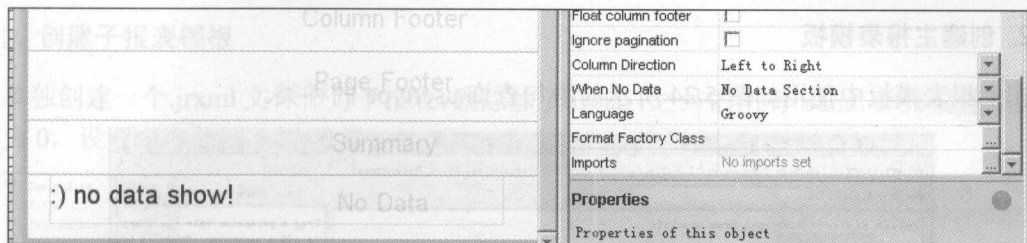


图 5.21 设置 When No Data 属性

再次将这两个.jrxml 放入 Web 项目，重新运行项目，主、子报表都显示出来了，效果如图 5.22 所示。

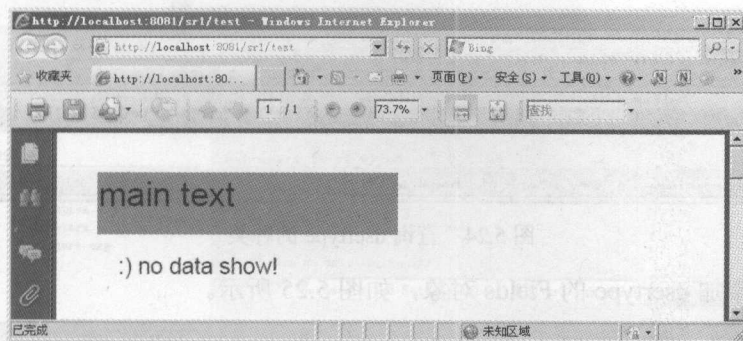


图 5.22 主、子报表都显示出来

5.1.3 子报表 Subreport 的示例——动态数据 JDBC

本小节将演示子报表的使用方法，数据源来自于 JDBC 的 Connection，重点是 Parameters 的使用，也就是 Subreport 控件 Parameters 属性中的 key/value 中的 key 要对应 sub.jrxml 中的同名 Parameters 对象，即把 main.jrxml 中的值通过 Subreport 控件的 Parameters 属性传递给 sub.jrxml 中的 Parameters 参数对象，因此 main.jrxml 和 sub.jrxml 就可以通信了。

1. 创建 userinfo 数据表及内容

设计数据表 userinfo 的内容，如图 5.23 所示。

TC03\SQL200... dbo. userinfo		TC03\SQL200... dbo. userinfo		
	id	username	password	usertype
▶	1	a1	aa	普通用户
	2	b1	bb	普通用户
	3	c1	cc	普通用户
	4	d2	dd	超级管理员
	5	e2	ee	超级管理员
	6	f3	ff	VIP用户
*	NULL	NULL	NULL	NULL

图 5.23 userinfo 数据表的内容

2. 创建主报表模板

在主报表模板中使用如图 5.24 所示的语句查询 usertype 的种类。

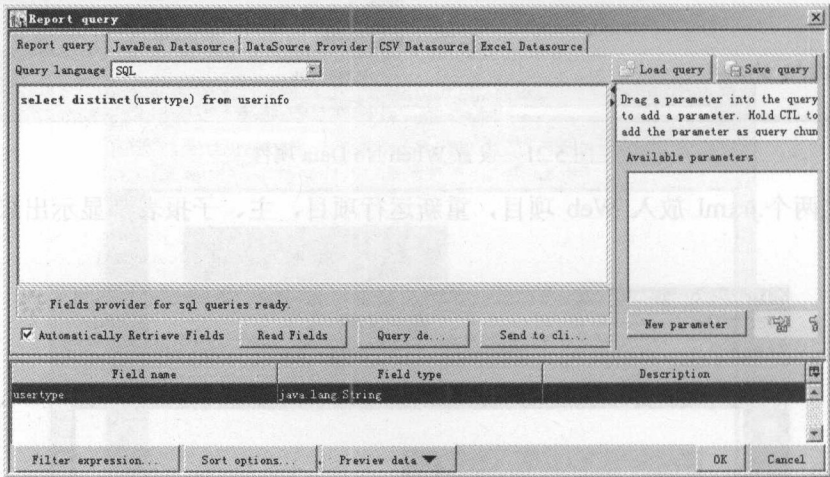


图 5.24 查询 usertype 的种类

向主报表中添加 usertype 的 Fields 对象，如图 5.25 所示。

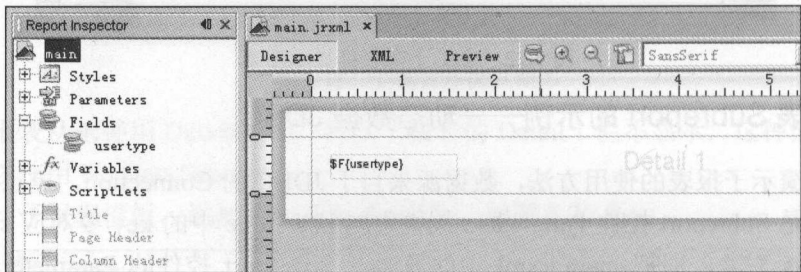


图 5.25 添加 usertype 的 Fields 对象

预览效果如图 5.26 所示，即出现 3 条记录。

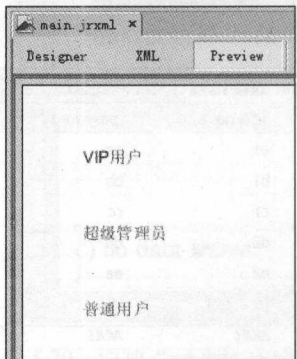


图 5.26 出现 3 条记录

3. 创建子报表模板

单独创建一个 jrxml 文件作为子报表, 删除除了 Detail 之外的所有 Band, 并且设置 Margins 属性为 0, 设置内容如图 5.27 所示。

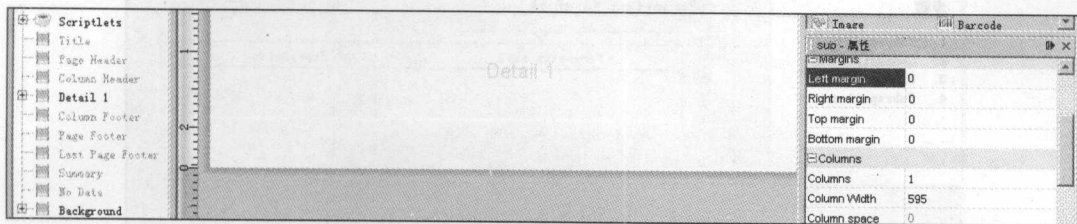


图 5.27 删除 Band 及设置 Margins 属性

返回主报表, 添加 Subreport 控件, 使用如图 5.28 所示的设置在 main.jrxml 中关联 sub.jrxml 文件。

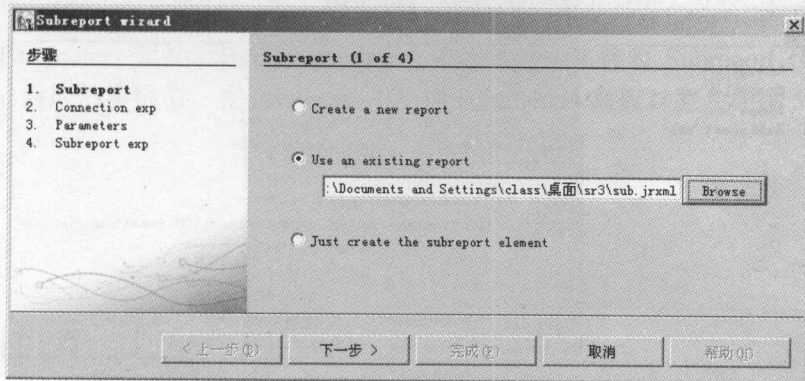


图 5.28 关联 sub.jrxml

单击“下一步”按钮, 配置主、子报表使用相同的数据源, 即同一个 Connection 连接, 如图 5.29 所示。

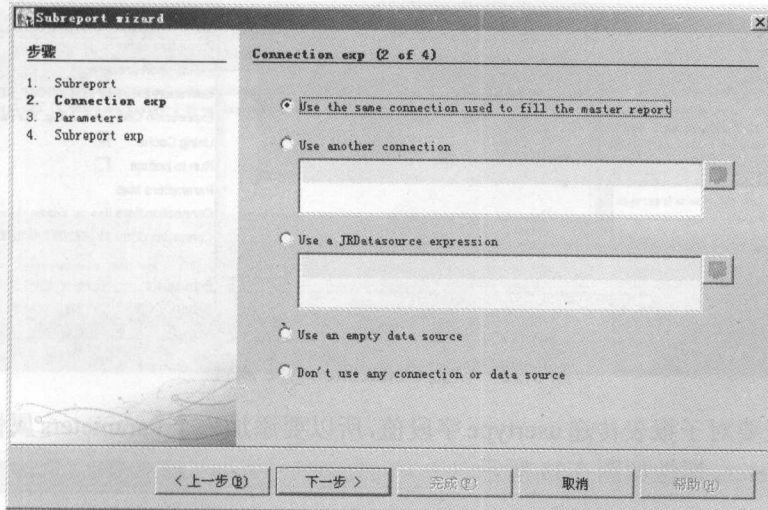


图 5.29 使用同一个 Connection 对象

单击“下一步”按钮，出现设置 Parameters 参数映射（如图 5.30 所示）和设置寻找.jasper 路径类型的对话框（如图 5.31 所示），保持默认值即可，无须配置。

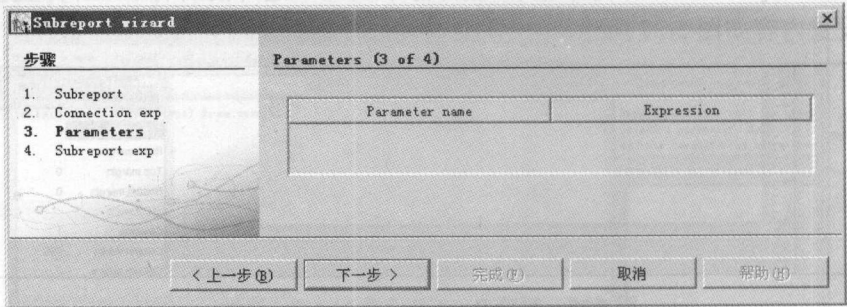


图 5.30 设置 Parameters 参数映射

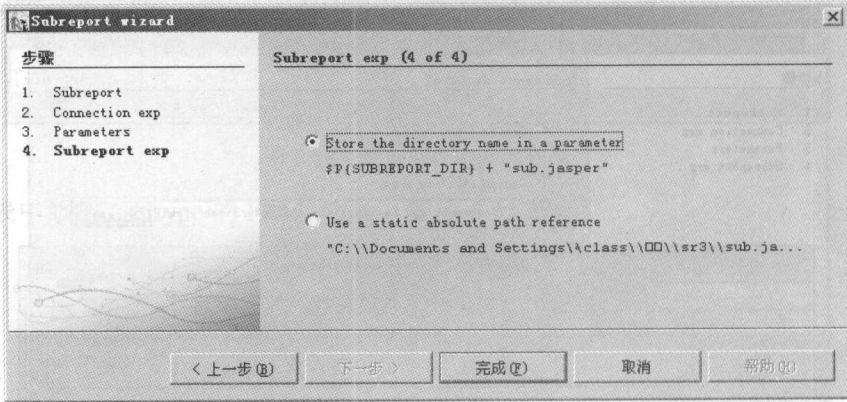


图 5.31 设置寻找.jasper 路径类型

单击“完成”按钮完成 Subreport 控件关联 sub.jrxml 的操作，还要配置 Subreport 控件的 Connection type 属性，如图 5.32 所示。

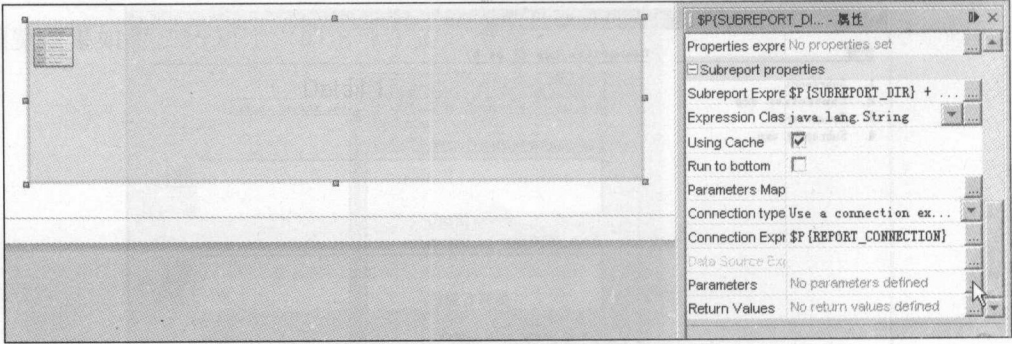


图 5.32 设置 Connection type 属性

由于主报表要对子报表传递 usertype 字段值，所以要添加一个 Parameters 属性，为 Subreport 控件添加 Parameters 属性如图 5.33 所示。

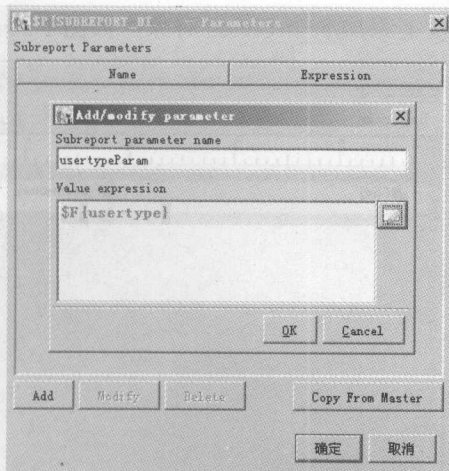


图 5.33 为 Subreport 控件添加 Parameters 属性

添加了 Parameters 属性后还要在 sub.jrxml 中添加参数来接收 Subreport 传递过来的参数 usertypeParam 对应的参数值，在 sub.jrxml 中添加 Parameters 参数对象，如图 5.34 所示。

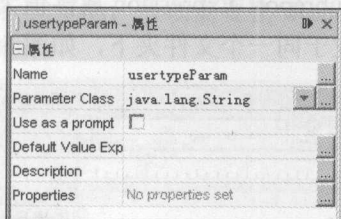


图 5.34 在 sub.jrxml 中添加 Parameters 参数对象

需要注意的是 Subreport 中的 Parameters 属性名 usertypeParam 一定要和 sub.jrxml 中定义的 Parameters 参数名 usertypeParam 相同，这样才可以正确传递数据。

在 sub.jrxml 中配置查询的 SQL 语句，如图 5.35 所示。

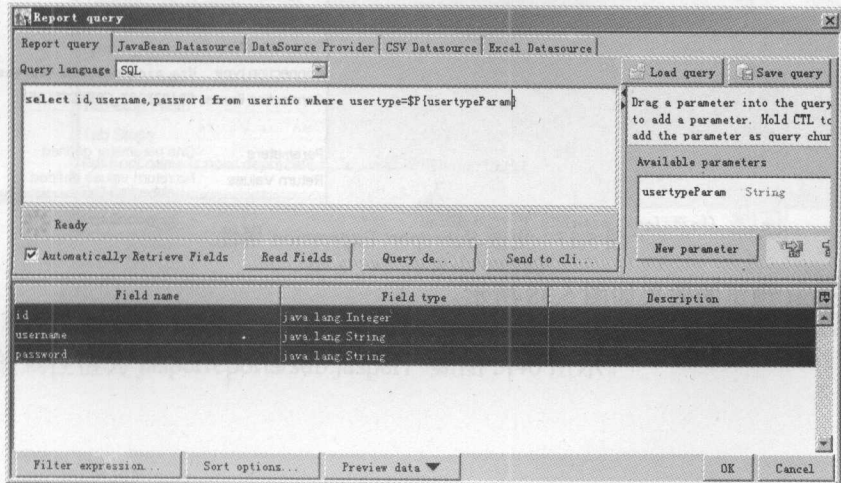


图 5.35 为 sub.jrxml 配置 SQL 语句

并且在 sub.jrxml 报表模板上添加 3 个 Fields 对象，如图 5.36 所示。

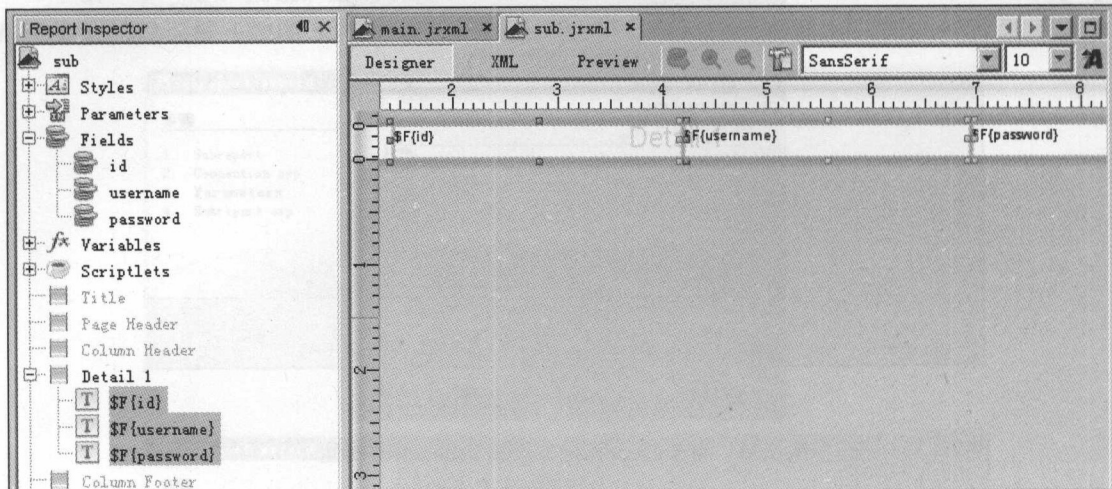


图 5.36 添加 3 个 Fields 对象

最后需要更改 Subreport 的 Subreport Expression 属性值为 sub.jasper，也就是在 iReport 中进行报表的预览，这两个 jrxml 位于同一个文件夹下，如图 5.37 所示。

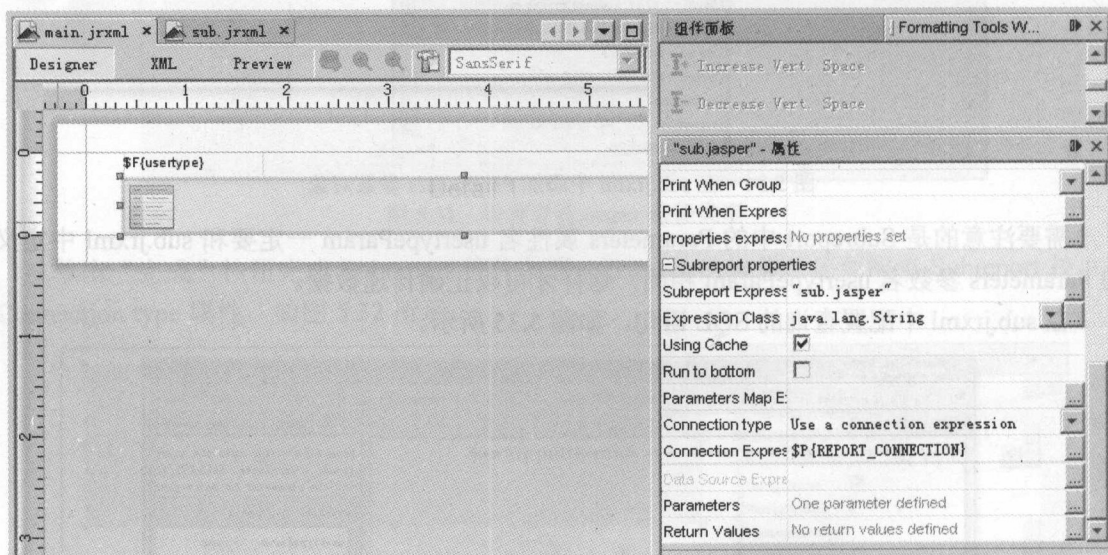


图 5.37 更改 Subreport Expression 属性

利用 iReport 预览的效果如图 5.38 所示。

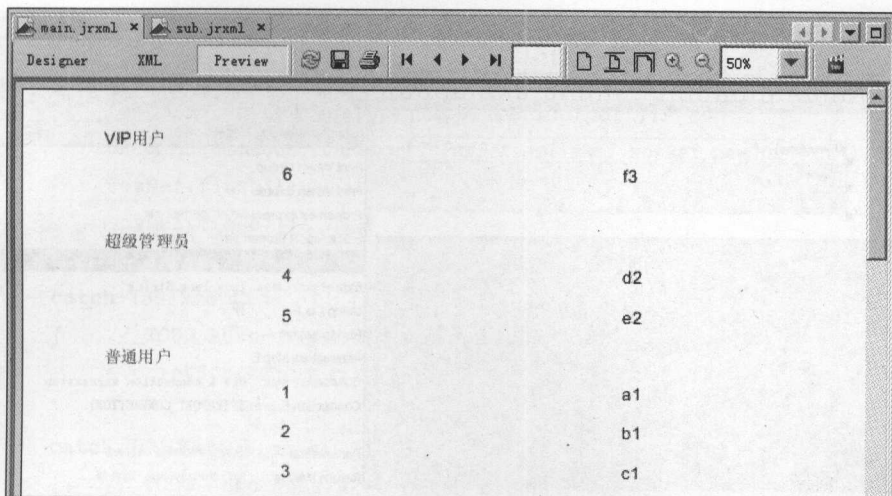


图 5.38 在 iReport 中的预览效果

4. 在 Web 项目中预览

由于 Web 项目是以 PDF 进行预览的，所以需要在 iReport 中解决 PDF 文件中显示中文的问题，设置中文编码如图 5.39 所示。

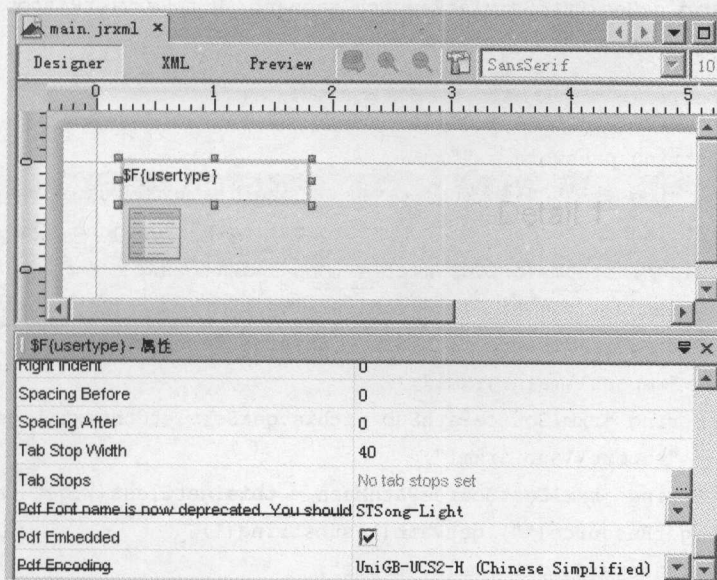


图 5.39 设置中文编码

另外，在 Web 项目中运行的 jasper 文件位于 jasperreports 包中，所以还要设置 Subreport Expression 的属性值为 jasperreports/sub.jasper，如图 5.40 所示。

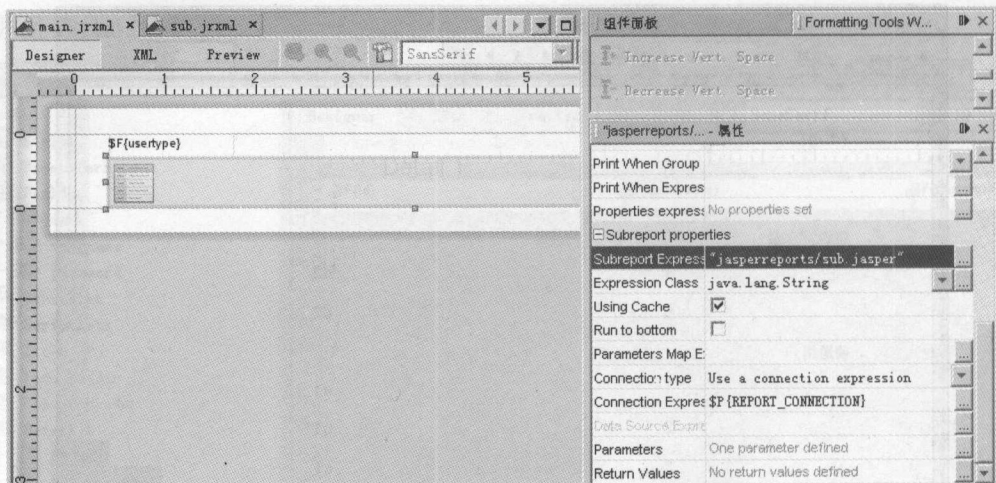


图 5.40 设置 Subreport Expression 属性值

把 iTextAsian.jar 和 iTextAsianCmaps.jar 复制到 Web 项目的 WEB-INF/lib 文件夹下。
创建 Servlet，核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            String username = "sa";
            String password = "";
            String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
            String driverName = "com.microsoft.sqlserver.jdbc.SQLServerDriver";
            Class.forName(driverName);
            Connection connection=DriverManager.getConnection(url,username,password);
            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
            + "report\\main.jrxml";
            String jrxmlSourcePathSub = this.getServletContext().getRealPath("/")
            + "report\\sub.jrxml";
            String jrxmlDestSourcePathMain = this.getClass().getClassLoader()
            .getResource("").getPath().substring(1)
            + "jasperreports/main.jasper";
            String jrxmlDestSourcePathSub = this.getClass().getClassLoader()
            .getResource("").getPath().substring(1)
            + "jasperreports/sub.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
            jrxmlDestSourcePathMain);
            JasperCompileManager.compileReportToFile(jrxmlSourcePathSub,
            jrxmlDestSourcePathSub);
        }
    }
}
```

```

InputStream isRef=new FileInputStream(new File(jrxmlDestSourcePathMain));
ServletOutputStream sosRef = response.getOutputStream();
response.setContentType("application/pdf");
JasperRunManager.runReportToPdfStream(isRef,sosRef,new HashMap(),connection);
sosRef.flush();
sosRef.close();
}
catch (JRException e)
{    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (SQLException e)
{    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (ClassNotFoundException e)
{    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

运行效果如图 5.41 所示。

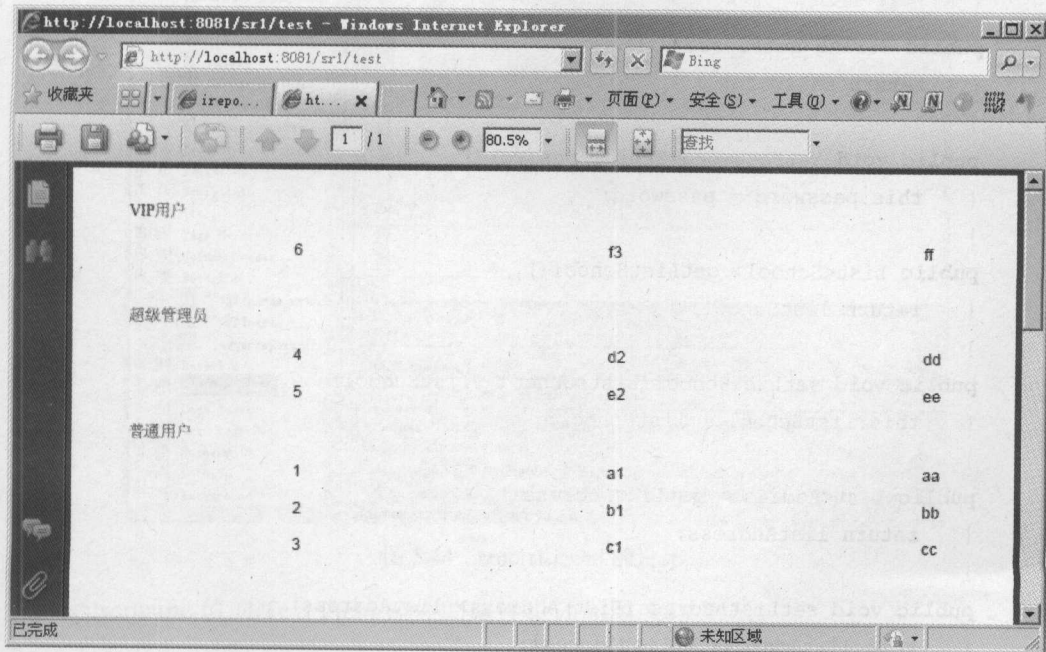


图 5.41 运行效果

5.1.4 子报表 Subreport 的示例——打印实体类中的 List<Userinfo>

在第 1 章中曾经介绍过将 List<Userinfo>中的数据打印到 Detail Band 中,但在有些情况下报表中的数据并不一定全是 List<Userinfo>列表,还可以打印一些如 username、password 等非集合类型的数据信息,即一个实体既有 String 属性,也有 List 属性,这样的情况正是使用子报表的最好时机。

1. 实体类的结构

创建要打印的实体 Userinfo.java,代码如下:

```
package entity;
import java.util.ArrayList;
import java.util.List;
public class Userinfo
{
    private String username;
    private String password;
    private List<School> listSchool = new ArrayList<School>();
    private List<Address> listAddress = new ArrayList<Address>();
    public String getUsername()
    {
        return username;
    }
    public void setUsername(String username)
    {
        this.username = username;
    }
    public String getPassword()
    {
        return password;
    }
    public void setPassword(String password)
    {
        this.password = password;
    }
    public List<School> getListSchool()
    {
        return listSchool;
    }
    public void setListSchool(List<School> listSchool)
    {
        this.listSchool = listSchool;
    }
    public List<Address> getListAddress()
    {
        return listAddress;
    }
    public void setListAddress(List<Address> listAddress)
    {
        this.listAddress = listAddress;
    }
}
```


类 Address.java 的结构如图 5.42 所示。

类 School.java 的结构如图 5.43 所示。

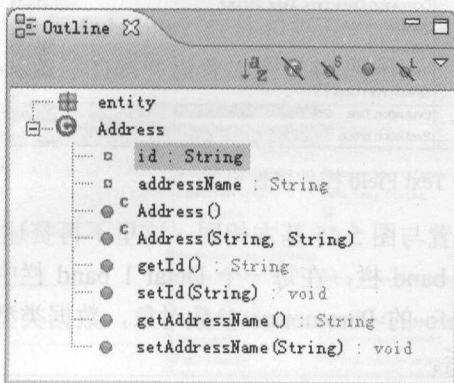


图 5.42 Address.java 类的结构

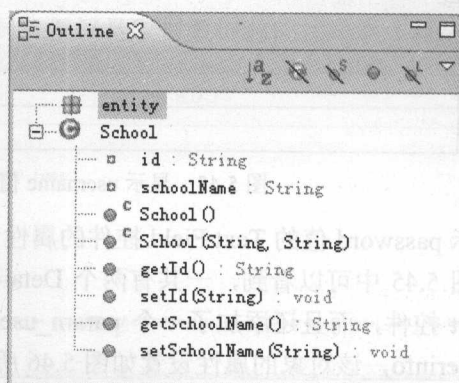


图 5.43 School.java 类的结构

2. 创建 3 个 .jrxml 报表模板文件

新建 report1.jrxml 报表模板，设计内容如图 5.44 所示。

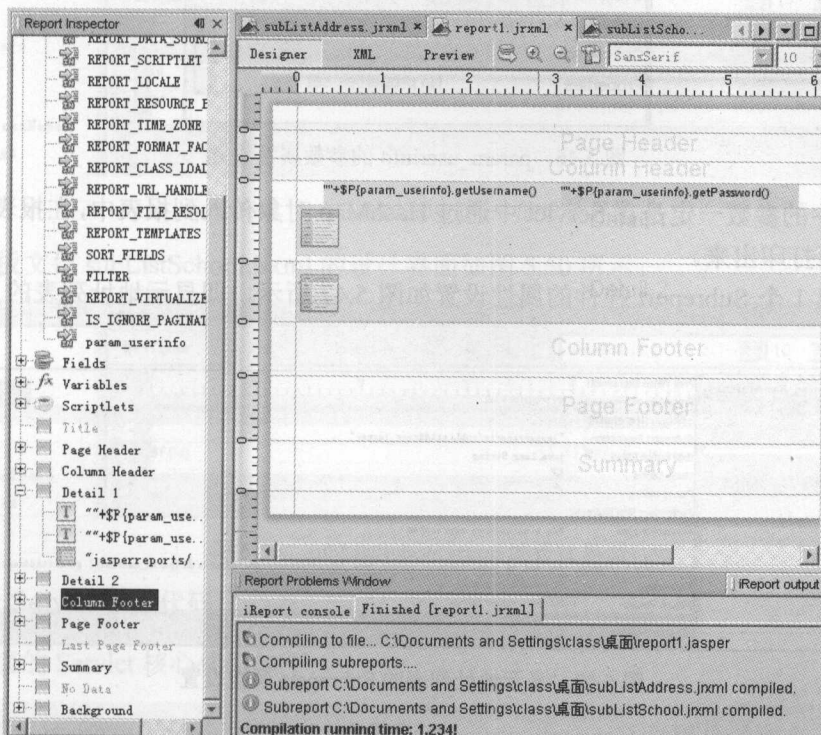


图 5.44 report1.jrxml 的设计

显示 username 值的 Text Field 控件的属性设计如图 5.45 所示。

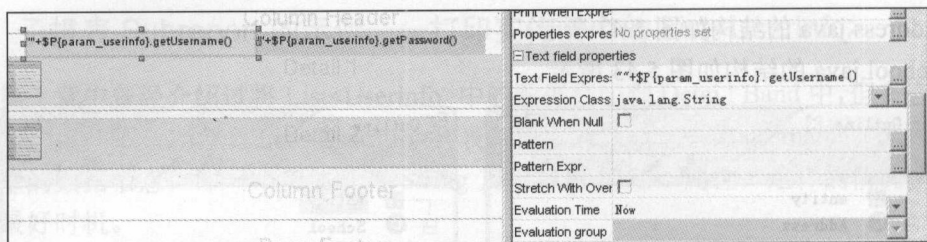


图 5.45 显示 username 值的 Text Field 控件属性设置

显示 password 值的 Text Field 控件的属性设置与图 5.45 基本相同, 这里不再赘述。

从图 5.45 中可以看到, 一共有两个 Detail band 栏, 在每一个 Detail band 栏中有一个 Subreport 控件, 而且还添加了一个 param_userinfo 的 Parameters 参数对象, 数据类型设置为 entity.Userinfo, 该对象的属性设置如图 5.46 所示。

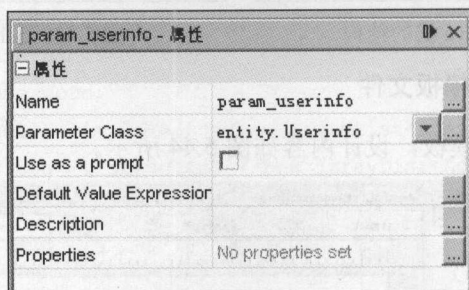


图 5.46 param_userinfo 的参数属性设置

图 5.46 中的参数一定是在 Servlet 中通过 HashMap 对象传入到报表中, 在报表中取得参数中的数据进而打印出来。

添加的第 1 个 Subreport 控件的属性设置如图 5.47 所示, 即显示地址列表的 Subreport 属性设置。

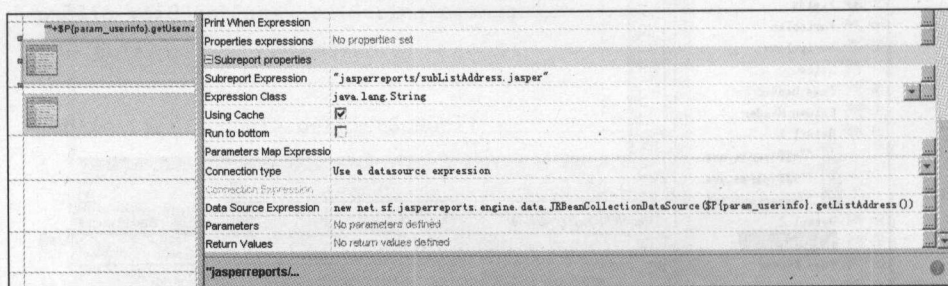


图 5.47 显示地址列表的 Subreport 属性设置

其中比较重要的是属性 Connection type, 设置为 Use a datasource expression, 也就是子报表的数据来自于一个数据源, 而不是 Connection JDBC 连接, 取得数据源 Data Source Expression 属性值的表达式的内容如下:

```
new net.sf.jasperreports.engine.data.
JRBeanCollectionDataSource($P{param_userinfo}.getListAddress())
```

即 Subreport 子报表的数据来自于 `$P{param_userinfo}` 参数的 `getListAddress()` 方法, 该方法返回一个 `List<Address>` 对象。

第 2 个 Subreport 控件和第 1 个基本相同, 仅仅是取得数据源的代码不一样, 第 2 个 Subreport 控件的属性设置如图 5.48 所示。

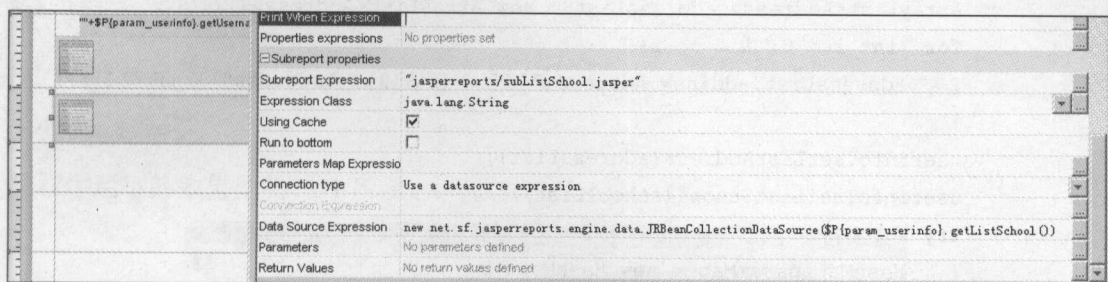


图 5.48 第 2 个 Subreport 控件的属性设置

报表模板文件 `subListAddress.jrxml` 的设计界面如图 5.49 所示。

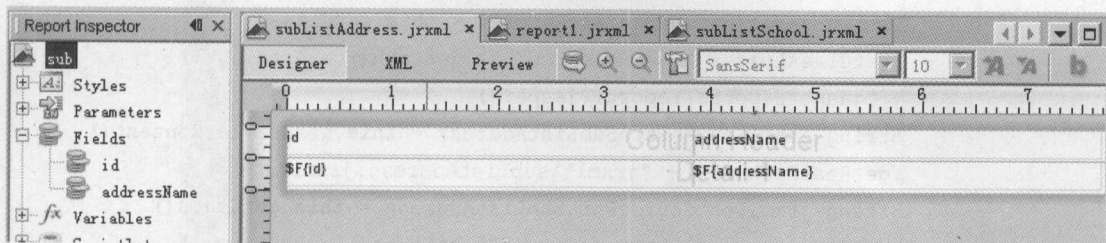


图 5.49 subListAddress.jrxml 的设计界面

报表模板文件 `subListSchool.jrxml` 的设计界面如图 5.50 所示。

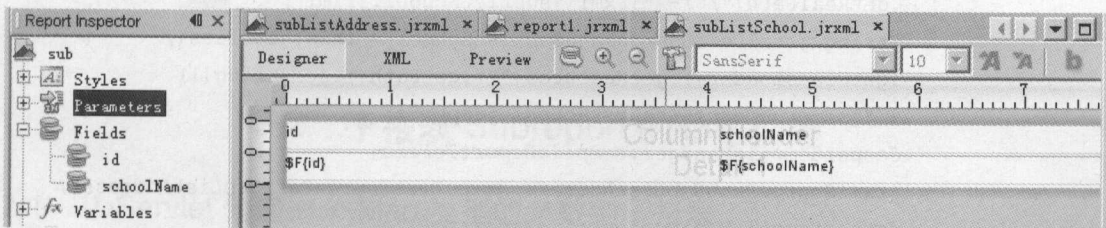


图 5.50 subListSchool.jrxml 的设计界面

3. 设计 Servlet 核心代码

运行报表的 Servlet 核心代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        Userinfo userinfo = new Userinfo();
        userinfo.setUsername("高洪岩账号");
        userinfo.setPassword("高洪岩密码");
    }
}
```



```

ArrayList<School> schoolList = new ArrayList<School>();
for (int i = 0; i < 3; i++)
{
    schoolList.add(new School("id" + (i + 1), "school" + (i + 1)));
}
ArrayList<Address> addressList = new ArrayList<Address>();
for (int i = 0; i < 3; i++)
{
    addressList.add(new Address("id" + (i + 1), "address" + (i + 1)));
}
userinfo.setListAddress(addressList);
userinfo.setListSchool(schoolList);
try
{
    HashMap paramMap = new HashMap();
    paramMap.put("param_userinfo", userinfo);
    String jrxmlSourcePathReport1=this.getServletContext().getRealPath("/")
    + "jrxml\\report1.jrxml";
    String jrxmlDestSourcePathReport1 = this.getClass()
    .getClassLoader().getResource("").getPath().substring(1)
    + "jasperreports/report1.jasper";
    String jrxmlSourcePathSubListAddress = this.getServletContext()
    .getRealPath("/") + "jrxml\\subListAddress.jrxml";
    String jrxmlDestSourcePathSubListAddress = this.getClass()
    .getClassLoader().getResource("").getPath().substring(1)
    + "jasperreports/subListAddress.jasper";
    String jrxmlSourcePathSubListSchool = this.getServletContext()
    .getRealPath("/") + "jrxml\\subListSchool.jrxml";
    String jrxmlDestSourcePathSubListSchool = this.getClass()
    .getClassLoader().getResource("").getPath().substring(1)
    + "jasperreports/subListSchool.jasper";
    JasperCompileManager.compileReportToFile(
    jrxmlSourcePathSubListAddress, jrxmlDestSourcePathSubListAddress);
    JasperCompileManager.compileReportToFile(
    jrxmlSourcePathSubListSchool, jrxmlDestSourcePathSubListSchool);
    JasperCompileManager.compileReportToFile(jrxmlSourcePathReport1,
    jrxmlDestSourcePathReport1);
    InputStream isRef = new FileInputStream(new File(
    jrxmlDestSourcePathReport1));
    ServletOutputStream sosRef = response.getOutputStream();
    response.setContentType("application/pdf");
    JasperRunManager.runReportToPdfStream(isRef, sosRef, paramMap,
    new JREmptyDataSource());
    sosRef.flush();
    sosRef.close();
}

```

```

    }
    catch (JRException e)
    {    // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

4. 运行效果

程序运行的效果如图 5.51 所示。

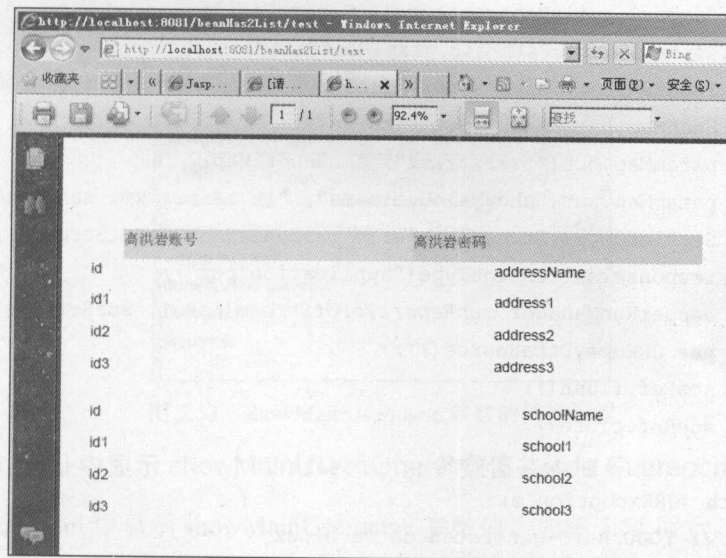


图 5.51 程序运行效果

5.2 子报表 Subreport 的参数传递

5.2.1 从 Servlet 传递一个 Map 类型的参数到子报表

使用 Parameters Map Expression 属性的情况是在 Servlet 中将固定的值作为参数传入到子报表中。

1. 创建 Servlet 代码

创建一个主报表 main.jrxml, 再创建一个子报表 sub.jrxml, Servlet 核心代码如下:

```

public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            HashMap subReportMap = new HashMap();

```



```

subReportMap.put("mapKey1", "i am mapKey1 Value!!!!");
String jrxmlSourcePathSub = this.getServletContext().getRealPath("/")
+ "jrxml\\sub.jrxml";
String jrxmlDestPathSub = this.getClass().getClassLoader()
.getResource("").getPath().substring(1)+ "jasperreports/sub.jasper";
String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
+ "jrxml\\main.jrxml";
String jrxmlDestPathMain = this.getClass().getClassLoader()
.getResource("").getPath().substring(1)+ "jasperreports/main.jasper";
JasperCompileManager.compileReportToFile(jrxmlSourcePathSub,jrxmlDestPathSub);
JasperCompileManager.compileReportToFile
(jrxmlSourcePathMain,jrxmlDestPathMain);
InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
HashMap paramMap = new HashMap();
paramMap.put("zzzzzzzzzz", subReportMap);
paramMap.put("showMainUsername", "in main.jrxml show username");
ServletOutputStream sosRef = response.getOutputStream();
response.setContentType("application/pdf");
JasperRunManager.runReportToPdfStream(isRef, sosRef, paramMap,
new JREmptyDataSource());
sosRef.flush();
sosRef.close();
}
catch (JRException e)
{
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}

```

从上面的代码可以看到如下的代码片段:

```

HashMap subReportMap = new HashMap();
subReportMap.put("mapKey1", "我是 mapKey1 我是高洪岩");
HashMap paramMap = new HashMap();
paramMap.put("zzzzzzzzzz", subReportMap);
paramMap.put("showMainUsername", "in main.jrxml show username");
JasperRunManager.runReportToPdfStream(isRef,sosRef,paramMap,new JREmptyDataSource());

```

对象 paramMap 是真正传给 main.jrxml 的 Parameters 参数的 Map 对象, 其中有两个 key, 一个是 zzzzzzzzzz, 另外一个为 showMainUsername, 所以要在 main.jrxml 中创建对应的两个 Parameters 参数对象。

2. 在 main.jrxml 中添加两个 Parameters 参数对象

参数名称为 zzzzzzzzzz 的属性设置如图 5.52 所示。

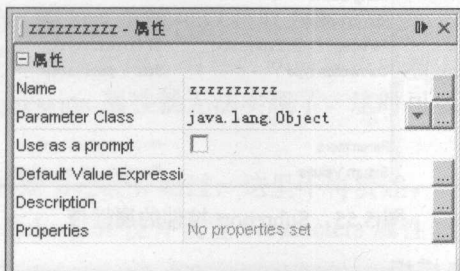


图 5.52 zzzzzzzzzz 参数的属性设置

参数 showMainUsername 的属性设置如图 5.53 所示。

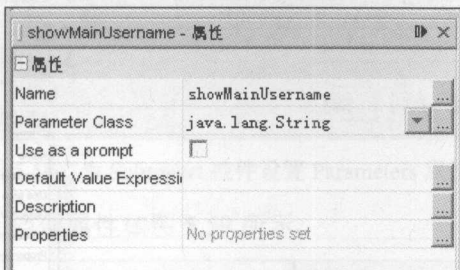


图 5.53 showMainUsername 参数的属性设置

3. 在 main.jrxml 中显示 showMainUsername 参数值并添加 Subreport 控件

设计在 main.jrxml 中显示 showMainUsername 参数值，如图 5.54 所示。

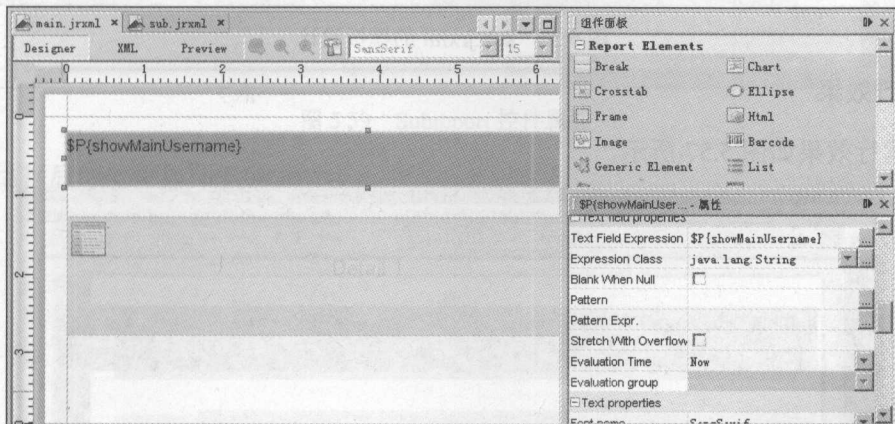


图 5.54 显示 showMainUsername 参数值

添加 Subreport 控件，并且设置其属性如图 5.55 所示。

Parameters Map Expression 的属性值为 `$P{zzzzzzzzzz}`，也就是将 `$P{zzzzzzzzzz}` 参数再传给子报表，所以在子报表中可以根据 Map 的 key 获取到 zzzzzzzzzz 对应 Map 中的值。

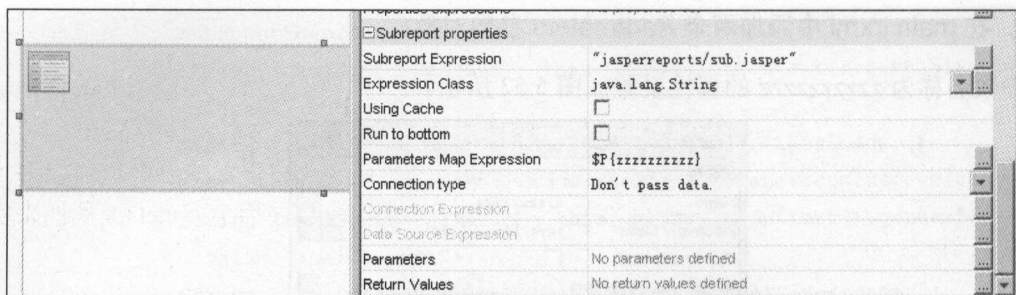


图 5.55 Subreport 控件的属性值

4. 设计 sub.jrxml 子报表模板

在子报表中添加 No Data Band，添加 Parameters 参数 mapKey1，效果如图 5.56 所示。

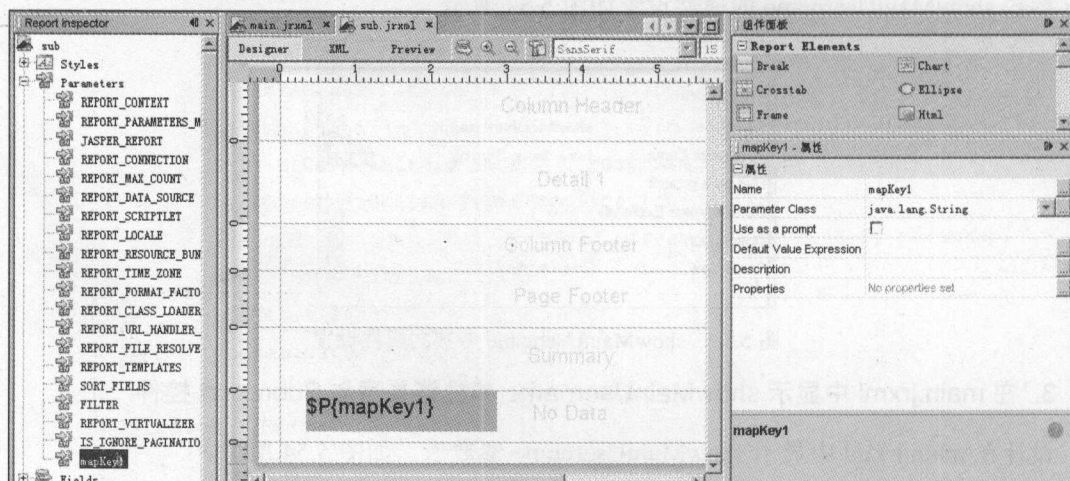


图 5.56 sub.jrxml 的模板设计

5. 运行效果

程序运行效果如图 5.57 所示。

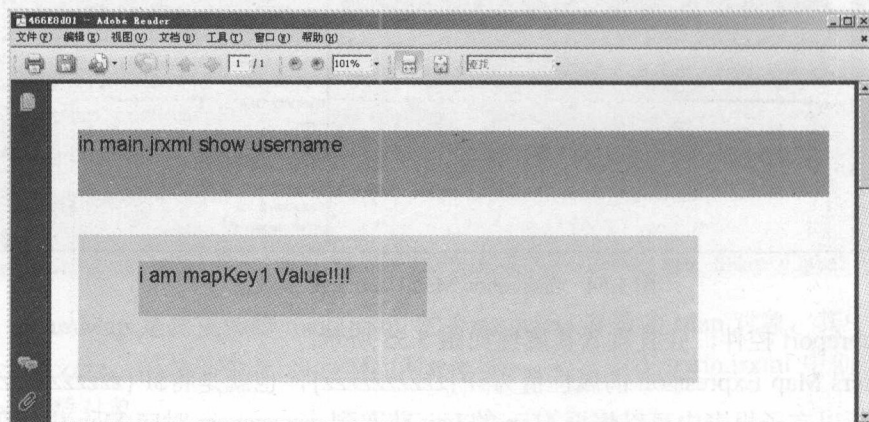


图 5.57 运行效果

5.2.2 对表达式进行计算后再传入子报表

前面的小节使用 Parameters Map Expression 属性对子报表传递 Map 类型的 Parameters 参数, 然后在子报表中根据 Map 的 key 值就可以获取到对应的值, 在这个过程中 Map 的值仅仅是从 Servlet 中就已经确定了, 不能更改, 有些时候需要传递到子报表中的参数值可以进行计算, 那么 Parameters Map Expression 属性就无法实现了, 这时可以使用 Parameters 属性来实现这种动态值的效果。

Parameters 属性在前面的章节已经介绍过, 这里不再赘述。

在上一小节的基础上对 Subreport 控件设置 Parameters 属性, 如图 5.58 所示。

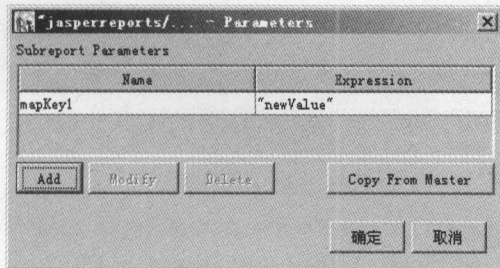


图 5.58 为 Subreport 控件设置 Parameters 属性

设置完成后的 Subreport 控件属性如图 5.59 所示。

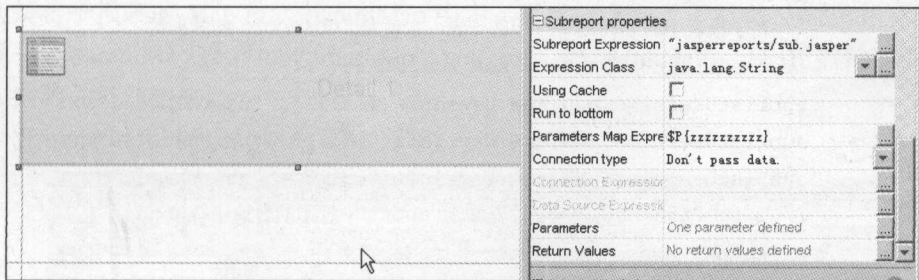


图 5.59 Subreport 控件属性

程序运行后的效果如图 5.60 所示。

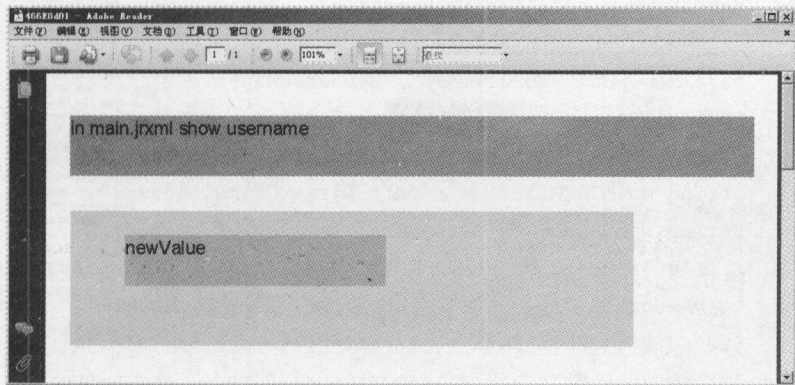


图 5.60 运行效果

从图 5.60 中可以看到, 通过使用 `Parameters` 属性可以对参数的值进行计算, 然后再传入到 Subreport 中。

5.2.3 对子报表传递 List<Userinfo>数据源

有时候子报表中的数据源是被嵌套的, 示例代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            // 在主报表中打印的普通 String 类型的 Parameters 参数
            HashMap paramMap = new HashMap();
            paramMap.put("param_username_main", "param_username_mainValue");
            paramMap.put("param_password_main", "param_password_mainValue");
            // 在主报表 Detail 打印的实体 Userinfo 列表
            List mainListUsername = new ArrayList();
            mainListUsername.add(new Userinfo("id11", "username11"));
            mainListUsername.add(new Userinfo("id12", "username12"));
            mainListUsername.add(new Userinfo("id13", "username13"));
            // //////////////////////////////////////
            // 在子报表中的 Detail 打印的列表, 也就是 Subreport 的数据源
            List subListUsername = new ArrayList();
            subListUsername.add(new Userinfo("id111", "username111"));
            subListUsername.add(new Userinfo("id112", "username112"));
            subListUsername.add(new Userinfo("id113", "username113"));
            // 此 subParamMap 是在子报表 Subreport 中打印的普通字符串
            HashMap subParamMap = new HashMap();
            subParamMap.put("param_username_sub", "param_username_subValue");
            subParamMap.put("param_password_sub", "param_password_subValue");
            // 将 subListUsername 放入 subParamMap 对象中
            subParamMap.put("subListUsername", subListUsername);
            // 再将 subParamMap 对象放入 paramMap 对象中
            paramMap.put("subParamMap", subParamMap);
            String reportsDir = request.getSession().getServletContext()
                .getRealPath("/reports");
            String mainJRXMLPath = reportsDir + "\\main.jrxml";
            String subJRXMLPath = reportsDir + "\\sub.jrxml";
            String jasperreportsPath = this.getClass().getClassLoader()
                .getResource("").getPath().toString().substring(1) + "/jasperreports/";
            String mainJASPERPath = jasperreportsPath + "main.jasper";
            String subJASPERPath = jasperreportsPath + "sub.jasper";
            JasperCompileManager.compileReportToFile(subJRXMLPath, subJASPERPath);
        }
    }
}
```

```

JasperCompileManager.compileReportToFile(mainJRXMLPath,mainJASPERPath);
InputStream isRef = new FileInputStream(new File(mainJASPERPath));
ServletOutputStream sosRef = response.getOutputStream();
response.setCharacterEncoding("utf-8");
response.setContentType("application/pdf");
JasperRunManager.runReportToPdfStream(isRef, sosRef, paramMap,
new JRBeanCollectionDataSource(mainListUsername));
sosRef.flush();
sosRef.close();
}
catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

从上面可以看到如下代码:

```
paramMap.put("subParamMap", subParamMap);
```

也就是将子报表的 Map 放入 paramMap 中了, 那也就意味着子报表 Subreport 的数据源的嵌套关系是: paramMap.get("subParamMap").get("subListUsername"), 只有这样才可以传递给子报表数据源并打印出来。

主报表 main.jrxml 的设计如图 5.61 所示。

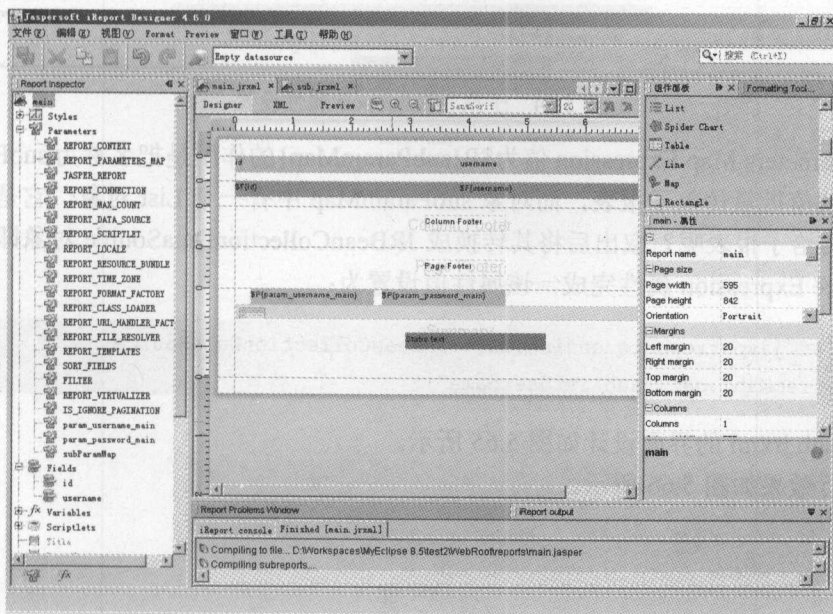


图 5.61 主报表 main.jrxml 的设计界面

从图 5.61 中可以看到有 3 个自定义的 Parameters 参数，属性设置如图 5.62 所示。

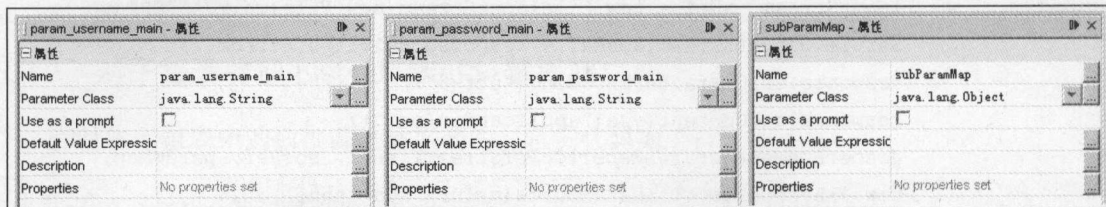


图 5.62 主报表 main.jrxml 中的 3 个 Parameters 对象属性设置

主报表 main.jrxml 中的 Summary 栏的内容如图 5.63 所示。

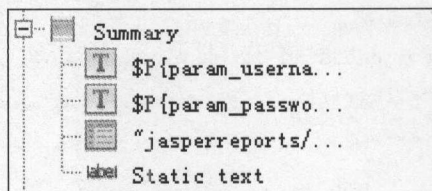


图 5.63 Summary 栏的内容

子报表 Subreport 控件的属性设置如图 5.64 所示。

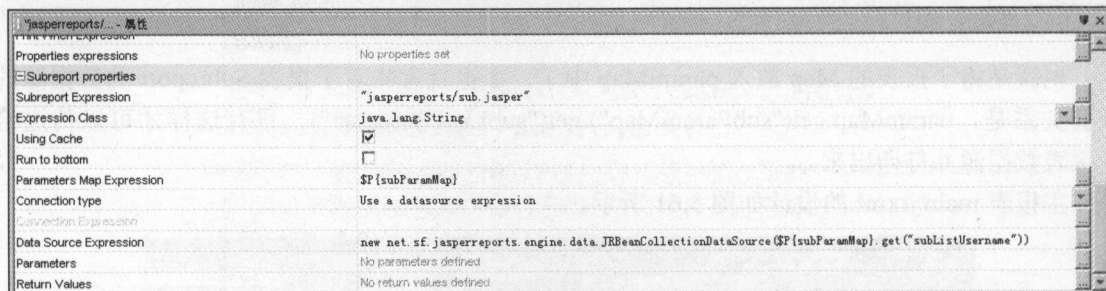


图 5.64 子报表 Subreport 属性设置

属性 Parameters Map Expression 值为 `$P{subParamMap}` 的作用是把名称为 `subParamMap` 的 Map 对象中的值批量传入子报表，而对象 `subParamMap` 中有一个 List 对象，它是子报表的数据源，如何传给子报表呢？取出后将其转换成 `JRBeanCollectionDataSource` 对象即可，此功能由 Data Source Expression 属性完成，该属性值设置为：

```
new net.sf.jasperreports.engine.data.JRBeanCollectionDataSource
($P{subParamMap}.get("subListUsername"))
```

子报表 sub.jrxml 的界面设计如图 5.65 所示。

程序运行效果如图 5.66 所示。

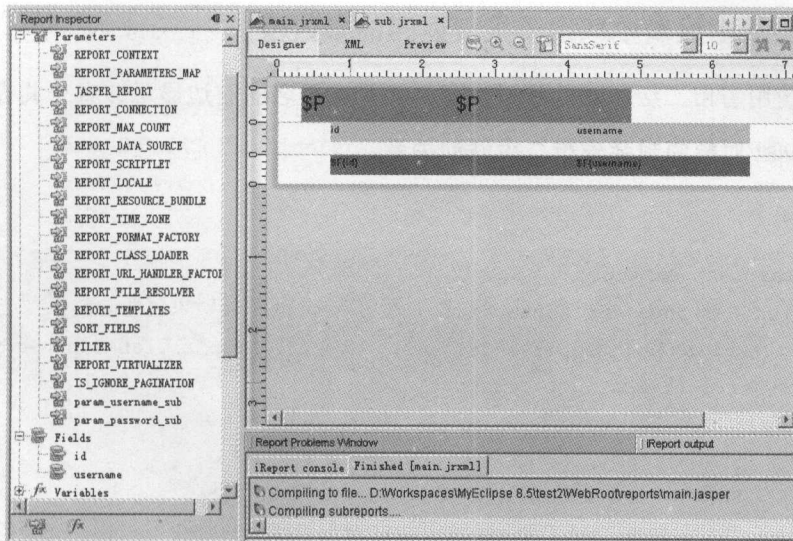


图 5.65 子报表的界面设计

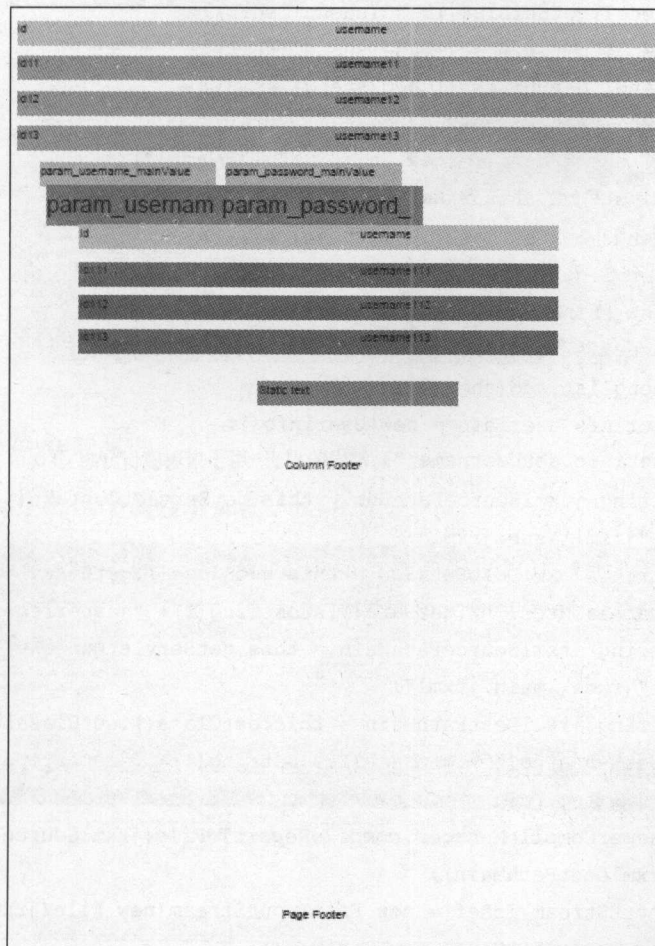


图 5.66 运行效果

5.2.4 示例：从主报表中取得子报表返回的参数值

本小节将使用省市二级联动的效果实现，并且在省名的右边显示该省一共有多少个市。

1. 在 Servlet 对象中封装省市二级联动关系

核心代码如下：

```
public class test extends HttpServlet
{
    @SuppressWarnings("unchecked")
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            Sheng sheng1 = new Sheng("1", "辽宁省");
            Shi sheng1_shi1 = new Shi("101", "沈阳市1");
            Shi sheng1_shi2 = new Shi("102", "沈阳市2");
            Shi sheng1_shi3 = new Shi("103", "沈阳市3");
            sheng1.getShiList().add(sheng1_shi1);
            sheng1.getShiList().add(sheng1_shi2);
            sheng1.getShiList().add(sheng1_shi3);
            Sheng sheng2 = new Sheng("2", "山东省");
            Shi sheng2_shi1 = new Shi("101", "青岛市1");
            Shi sheng2_shi2 = new Shi("101", "青岛市2");
            sheng2.getShiList().add(sheng2_shi1);
            sheng2.getShiList().add(sheng2_shi2);
            List shengList = new ArrayList();
            shengList.add(sheng1);
            shengList.add(sheng2);
            Userinfo userinfo = new Userinfo();
            userinfo.setUsername("高洪岩的报表-用子报表模仿分组 Group 的效果");
            String jrxmlSourcePathSub = this.getServletContext().getRealPath("/")
                + "jrxml\\sub.jrxml";
            String jrxmlDestPathSub = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1) + "jasperreports/sub.jasper";
            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
                + "jrxml\\main.jrxml";
            String jrxmlDestPathMain = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1) + "jasperreports/main.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePathSub, jrxmlDestPathSub);
            JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
                jrxmlDestPathMain);
            InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
            HashMap paramMap = new HashMap();
            paramMap.put("userinfo", userinfo);
```

```

ServletOutputStream sosRef = response.getOutputStream();
response.setContentType("application/pdf");
JasperRunManager.runReportToPdfStream(isRef, sosRef, paramMap,
new JRBeanCollectionDataSource(shengList));
sosRef.flush();
sosRef.close();
}
catch (JRException e)
{
// TODO Auto-generated catch block
e.printStackTrace();
}
}
}

```

2. 添加 Sheng.class、Shi.class 及中文.jar 文件路径

配置 Sheng.class、Shi.class 及中文.jar 的文件路径如图 5.67 所示。

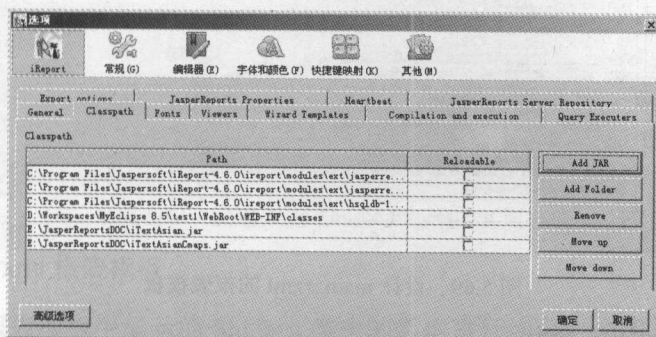


图 5.67 配置路径

3. 逆向 Sheng.java 的属性

逆向 Sheng.java 中的两个属性名如图 5.68 所示。

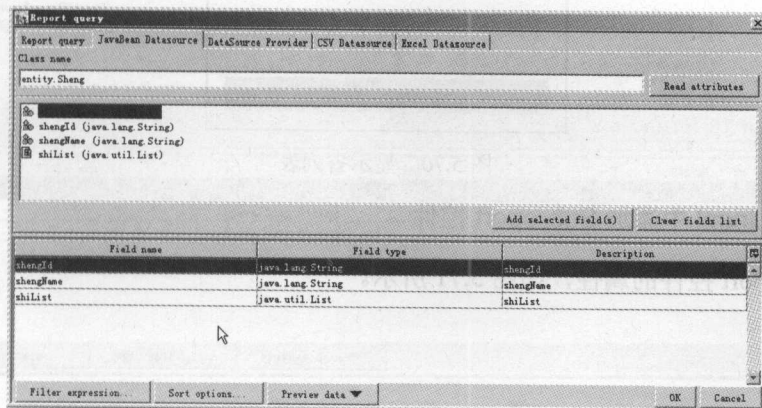


图 5.68 逆向出 Sheng.java 中的两个属性名

一定要将 shiList 属性逆向, 因为需要对 Shi.java 进行列表打印。

4. 设计 main.jrxml 的报表模板

main.jrxml 报表模板的外观如图 5.69 所示。

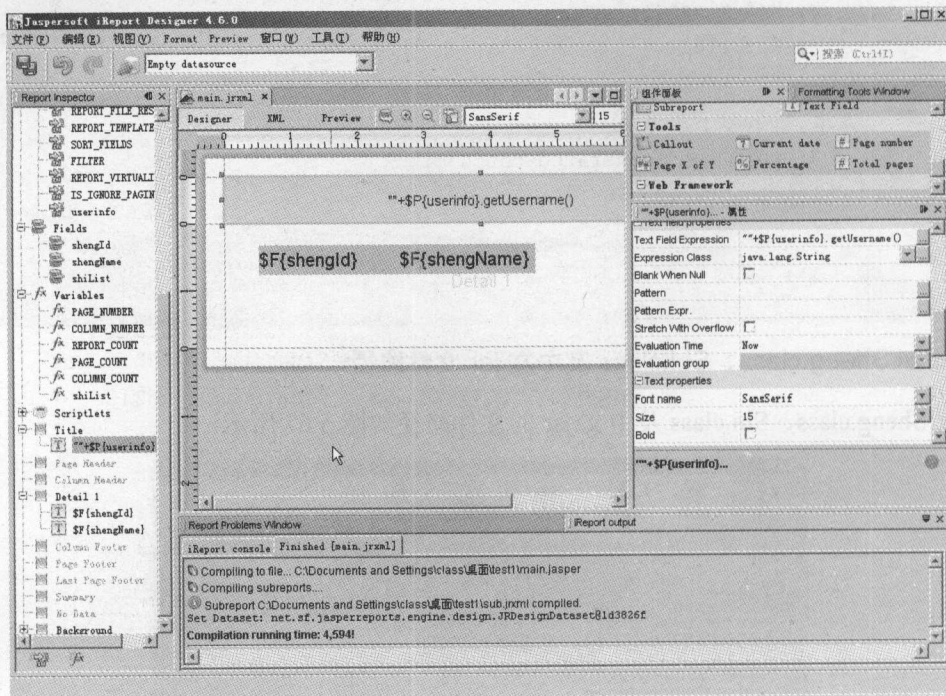


图 5.69 设计 main.jrxml 的报表模板

5. 显示省列表的运行效果

程序运行后显示出省列表, 如图 5.70 所示。

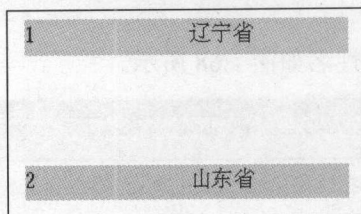


图 5.70 显示省列表

6. 在 main.jrxml 中添加 Subreport 控件

设置 Subreport 控件的属性, 如图 5.71 所示。

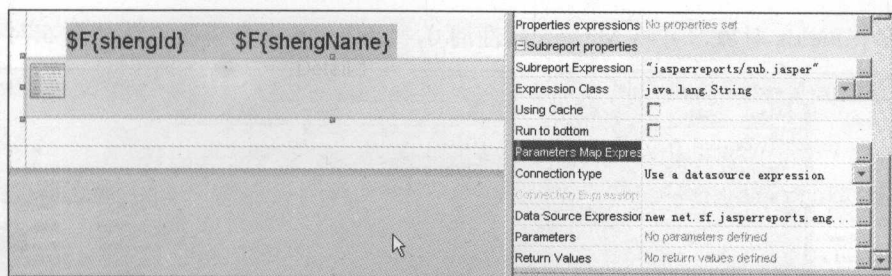


图 5.71 设置 Subreport 控件属性

关键是要设置 Data Source Expression 属性值, 如图 5.72 所示。

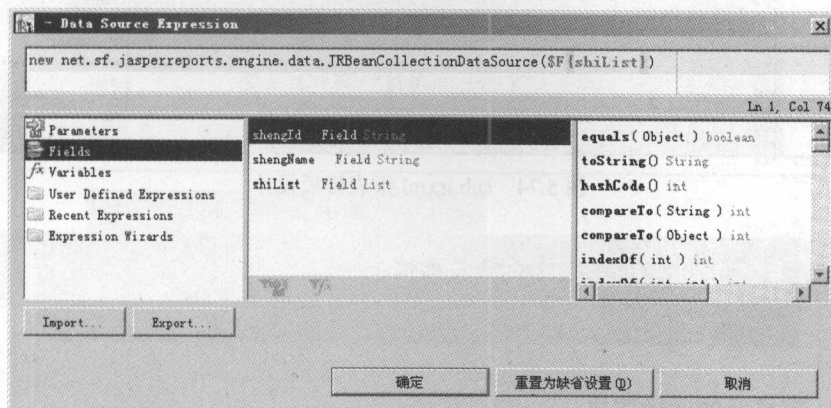


图 5.72 设置 Data Source Expression 属性值

7. 设计 sub.jrxml 报表模板

为 sub.jrxml 报表模板中显示的 Shi.java 类中的数据进行逆向, 如图 5.73 所示。

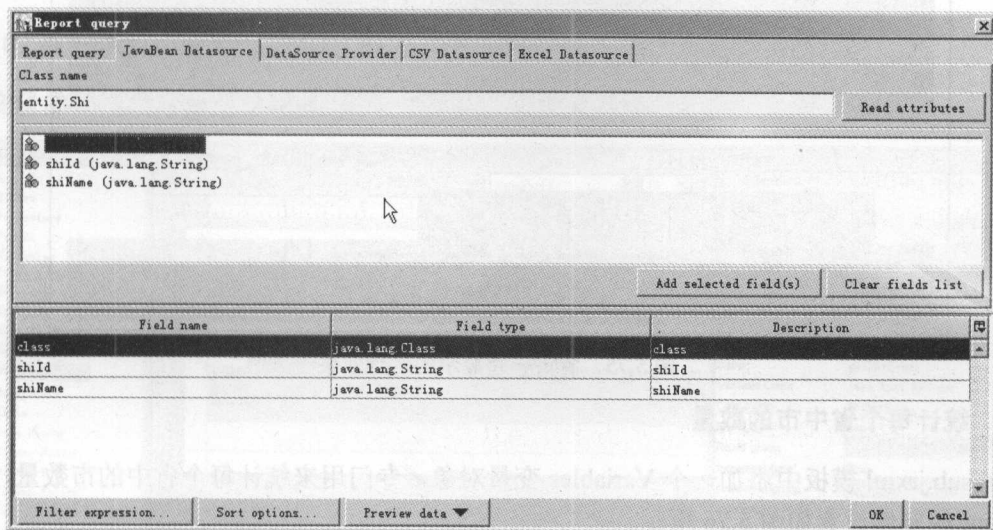


图 5.73 逆向 Shi.java 类

添加 3 个 Fields 对象，并将 Margin 属性清 0，sub.jrxml 报表模板设计如图 5.74 所示。

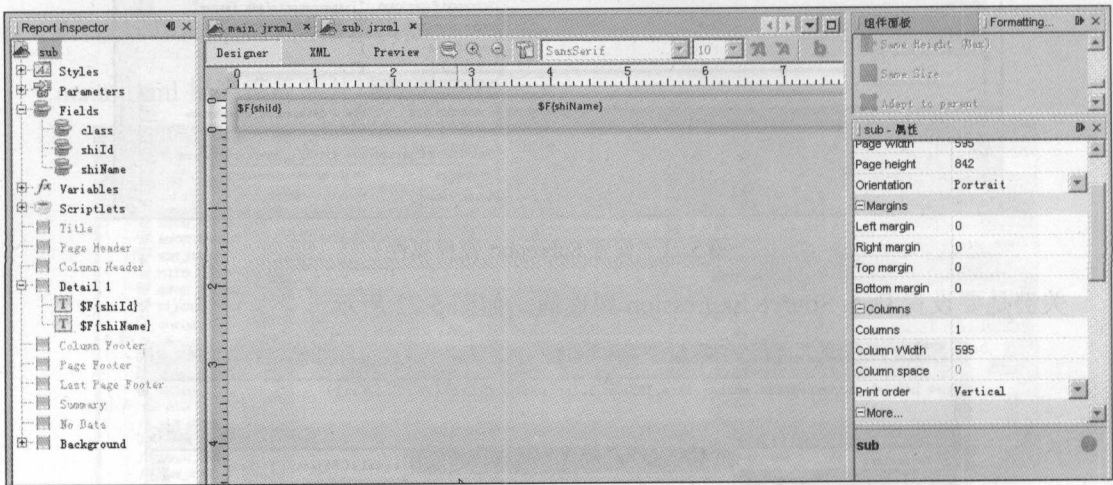


图 5.74 sub.jrxml 报表模板设计



别忘了设置 Text Field 控件的中文属性。

提示

8. 省市一起显示的运行效果

省市一起显示的运行效果如图 5.75 所示。

1		辽宁省	
101		沈阳市1	
102		沈阳市2	
103		沈阳市3	
2		山东省	
101		青岛市1	
101		青岛市2	

图 5.75 省市一起显示的运行效果

9. 统计每个省中市数量

在 sub.jrxml 模板中添加一个 Variables 变量对象，专门用来统计每个省中的市数量，变量 shiCount 的属性设置如图 5.76 所示。

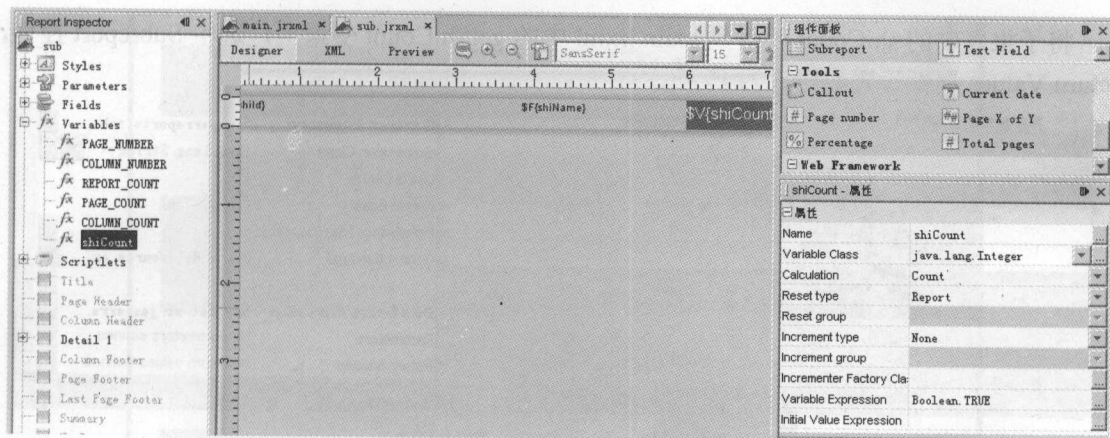


图 5.76 变量 shCount 的属性设置

分别显示出市的数量，程序运行效果如图 5.77 所示。

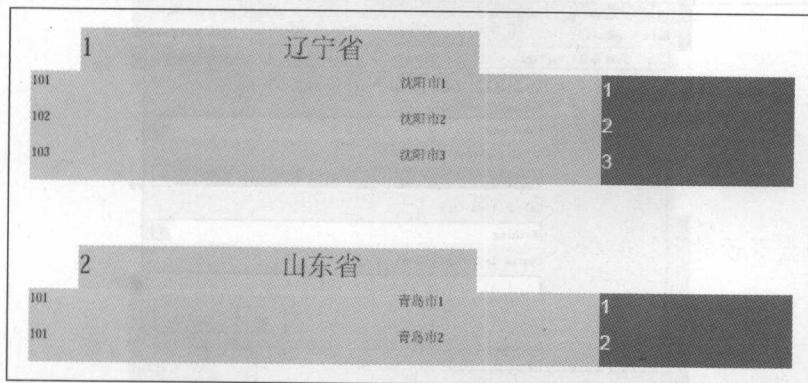


图 5.77 分别显示出市的数量

10. 在 main.jrxml 中添加 Variables 变量并与子报表中的变量做关联映射

变量的属性设置如图 5.78 所示。

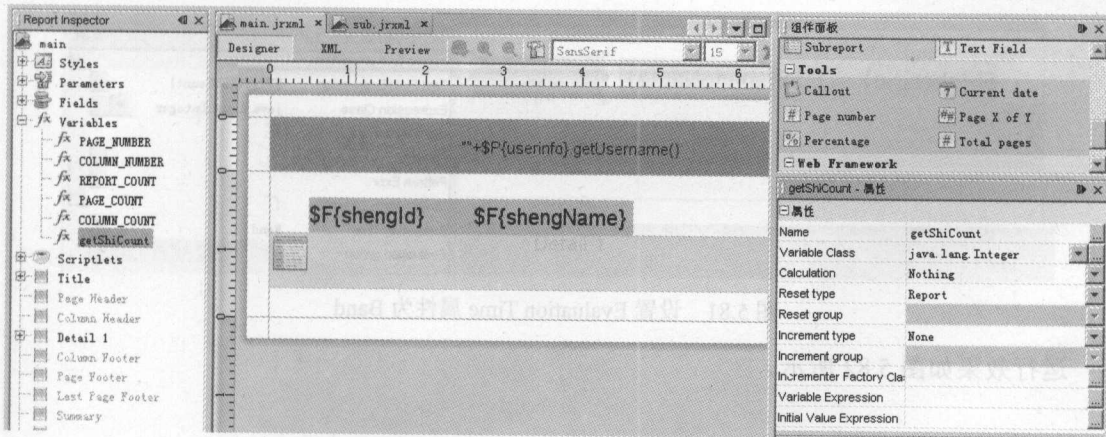


图 5.78 main.jrxml 中的变量设置

将子报表中的 shiCount 映射到 main.jrxml 中的 getShiCount 变量上, 选中 Subreport 控件; Return Values 属性设置如图 5.79 所示。

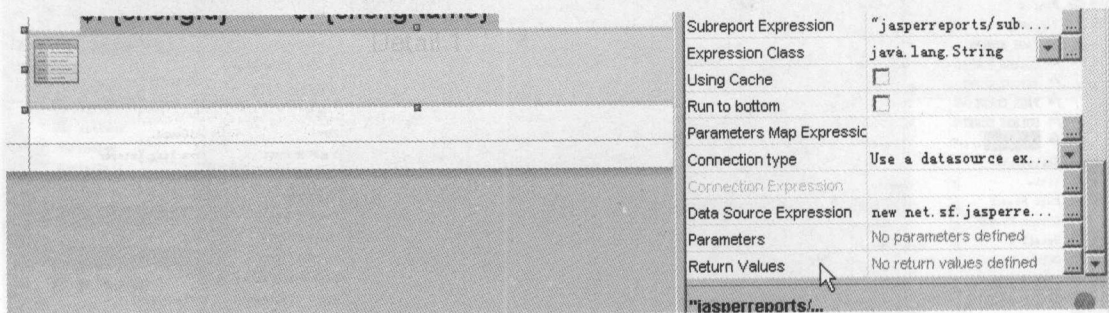


图 5.79 Return Values 属性设置

添加一个变量映射, 主、子报表的变量映射设置如图 5.80 所示。

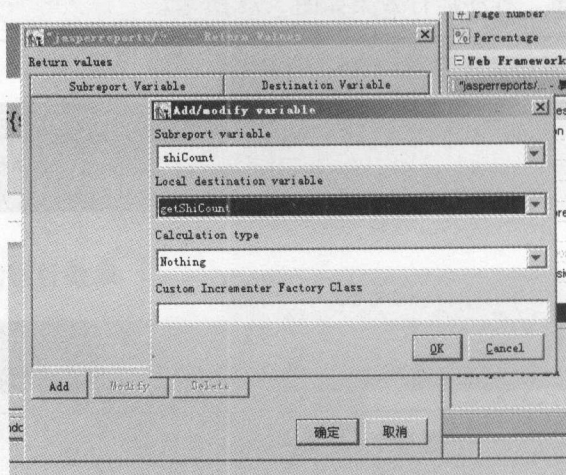


图 5.80 设置主、子报表的变量映射

将变量添加到 main.jrxml 界面, 并且设置 Evaluation Time 属性为 Band, 如图 5.81 所示。

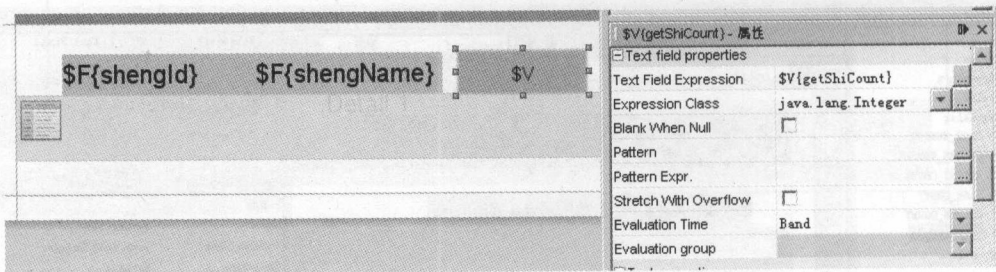


图 5.81 设置 Evaluation Time 属性为 Band

运行效果如图 5.82 所示。

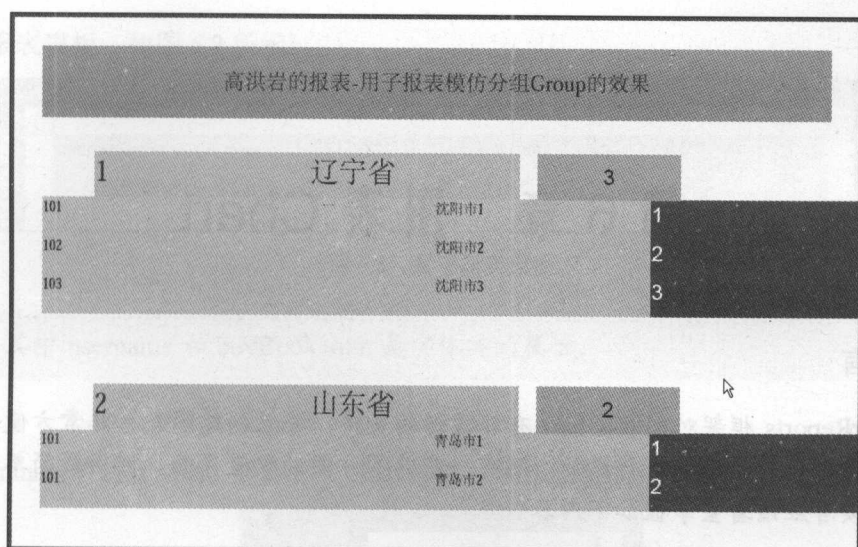


图 5.82 最终运行效果

6.1 图表 Chart 的使用——饼状图

饼图是报表中最常用的图表之一，它主要用于显示数据的比例关系。在报表中，饼图可以直观地展示某一数据项在整体中的占比。例如，在销售报表中，饼图可以用来展示不同产品类别的销售占比。饼图的制作相对简单，只需要将数据源中的某一列数据作为饼图的数据源即可。在报表设计中，饼图通常与表格结合使用，以便用户能够更直观地了解数据的分布情况。

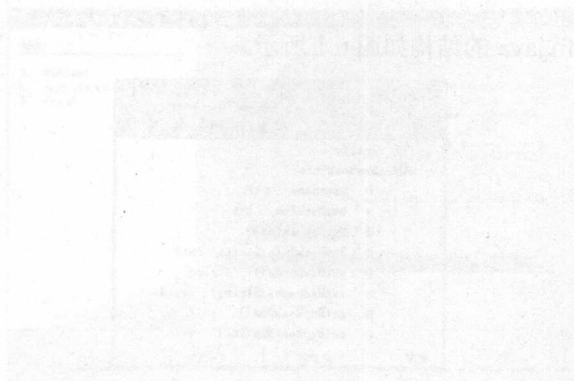


图 6.1 新建 JavaBean 数据源的报表模板

第 6 章 图表 Chart

↓ 导言

JasperReports 框架对图表 Chart 有非常好的支持, 并且创建图表也非常方便, 以所见即所得及属性面板的方式进行设计, 还有一些非常方便的配置界面, 可用最简单的步骤创建图表, 读者应该着重掌握如下内容:

- ★ 饼状、柱状、曲线图的使用
- ★ Chart 控件的属性
- ★ 图表 Chart 结合 SQL 及 JavaBean 数据源的使用
- ★ 为图表 Chart 添加超链接
- ★ 皮肤在图表中的使用

6.1 图表 Chart 的使用——饼状图

前面的章节都是在报表模板中使用文字式的报表, 如果使用图形式报表作为数据的展示将会非常直观和新颖, 在 JasperReports 框架中对图表 Chart 的支持是通过 JFreeChart 框架实现的, 其与数据源联合使用比较方便, 在本章中将详细为大家介绍在 iReport 中使用 Chart 的方法。

在控件 Chart 中展示的数据来源可以是 Main dataset 主数据集, 也可以是 Sub dataset 子数据集, 有关更加具体的 Subdataset 子数据集的使用请参考后面的章节。

6.1.1 新建 JavaBean 数据源的报表模板

实体类 BuyBookInfo.java 的结构如图 6.1 所示。

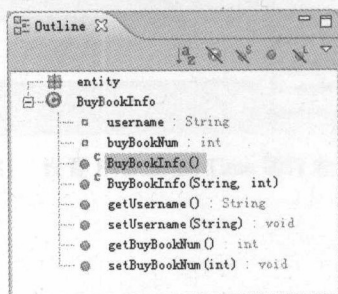


图 6.1 BuyBookInfo.java 类结构

新建报表模板，如图 6.2 所示。

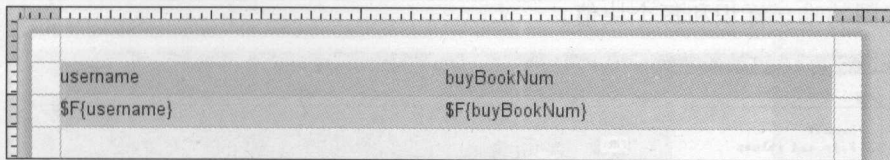


图 6.2 新建报表模板



其中 username 和 buyBookNum 是实体类的属性。

提示

对 Summary 栏进行加高，然后放入 Chart 控件，弹出选择图表样式的对话框，如图 6.3 所示。

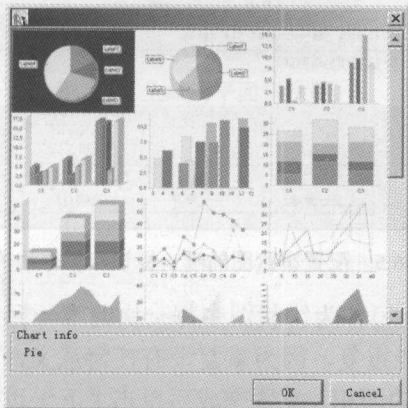


图 6.3 选择图表样式

在图 6.3 中选择第 1 个选项，即平面饼状图，也就是 Pie 图表样式。

6.1.2 配置 Chart

在图 6.3 中单击 OK 按钮完成对 Chart 控件的添加，弹出如图 6.4 所示的选择数据源配置对话框。

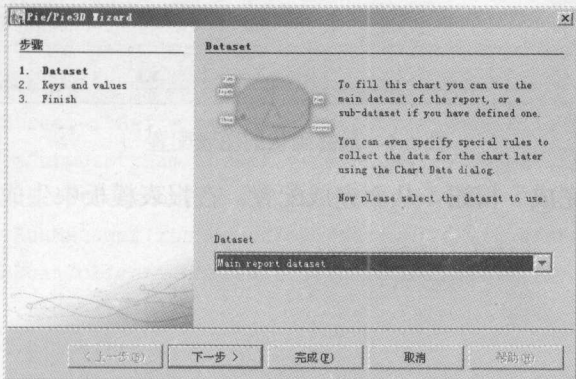


图 6.4 选择数据源

在图 6.4 中选择一个数据源, 单击“下一步”按钮继续配置, 弹出如图 6.5 所示的对话框, 即配置饼状图的唯一标识和每个块值。

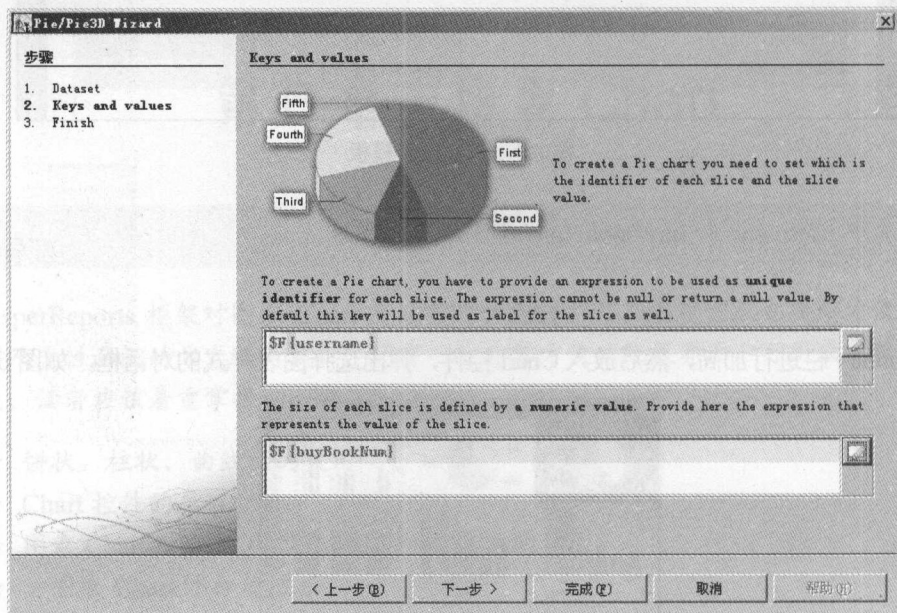


图 6.5 配置饼状图的唯一标识和每个块值

在图 6.5 中使用 `$F{username}` 作为饼状图中每一块的唯一标识, 这就要求 `List<JavaBean>` 数据源中的 `username` 属性不能重复, 因为重复没有任何意义, 下面使用 `$F{buyBookNum}` 来定义每一块在饼状图中所占用的面积, 配置完成后单击“下一步”按钮继续配置, 最终确认的图表配置如图 6.6 所示。

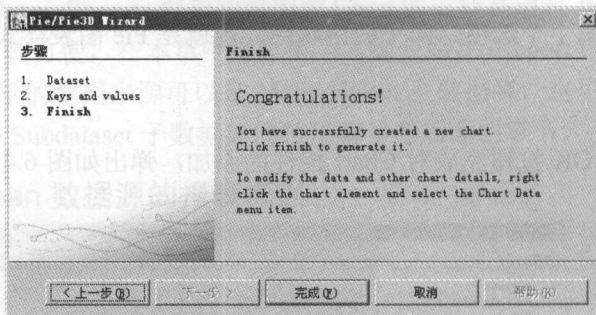


图 6.6 最终确认的图表配置

在图 6.6 中单击“完成”按钮, 从而完成配置, 在报表模板中生成一个饼状图, 如图 6.7 所示。

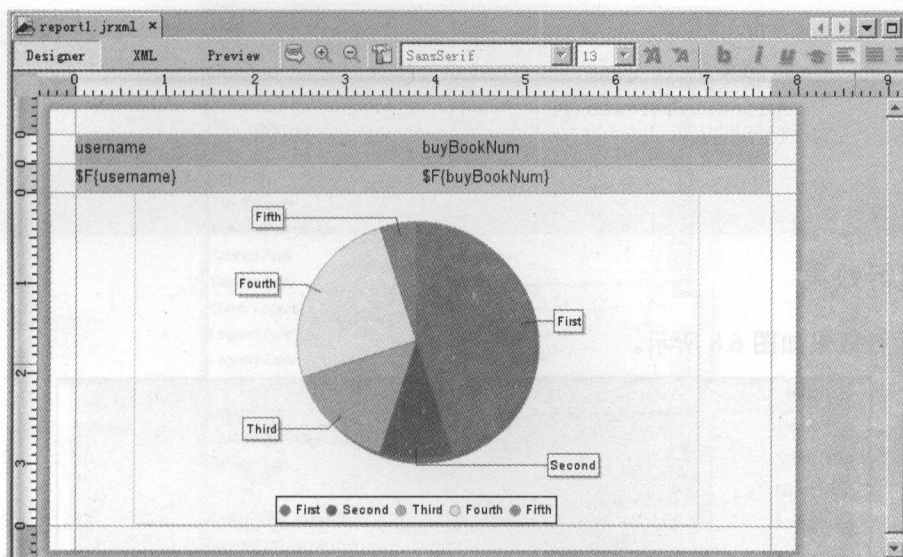


图 6.7 生成的平面饼状图

6.1.3 创建 Servlet 对象

创建的 Servlet 对象核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List buyBookInfoList = new ArrayList();
            for (int i = 0; i < 10; i++)
            {
                buyBookInfoList.add(new BuyBookInfo("username" + (i + 1), (i + 1)));
            }

            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
                + "jrxml\\report1.jrxml";
            String jrxmlDestPathMain = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1) + "jasperreports/report1.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
                jrxmlDestPathMain);
            InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
            ServletOutputStream sosRef = response.getOutputStream();
            response.setContentType("application/pdf");
            JasperRunManager.runReportToPdfStream(isRef, sosRef, new HashMap(),
                new JRBeanCollectionDataSource(buyBookInfoList));
            sosRef.flush();
            sosRef.close();
        }
    }
}
```

```

catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}

```

6.1.4 运行效果

程序运行效果如图 6.8 所示。

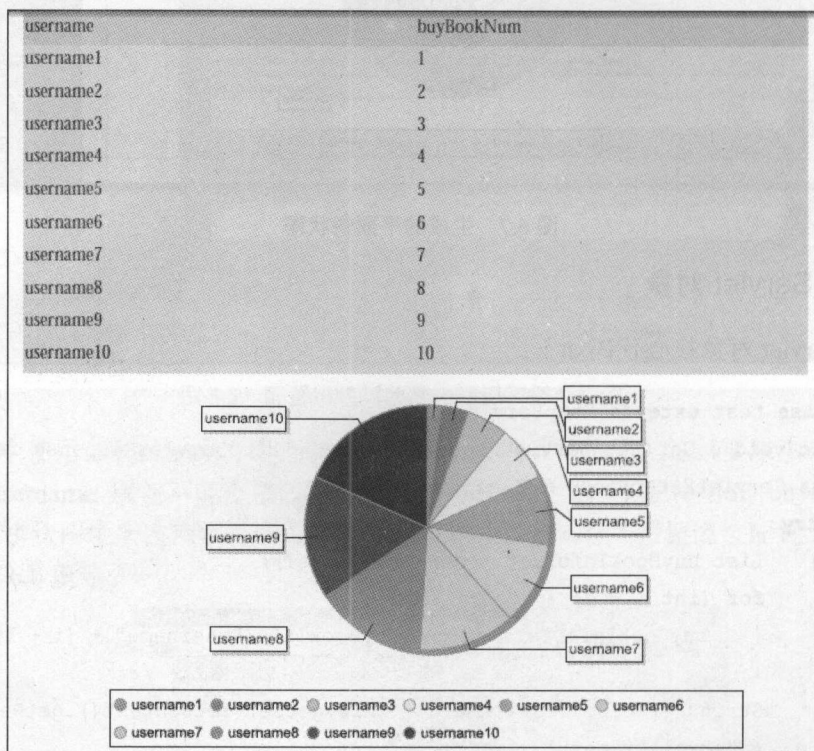


图 6.8 运行效果

6.1.5 图表 Chart 的常用属性——饼状图

图表 Chart 有一些自己独有的属性，如图 6.9 所示。

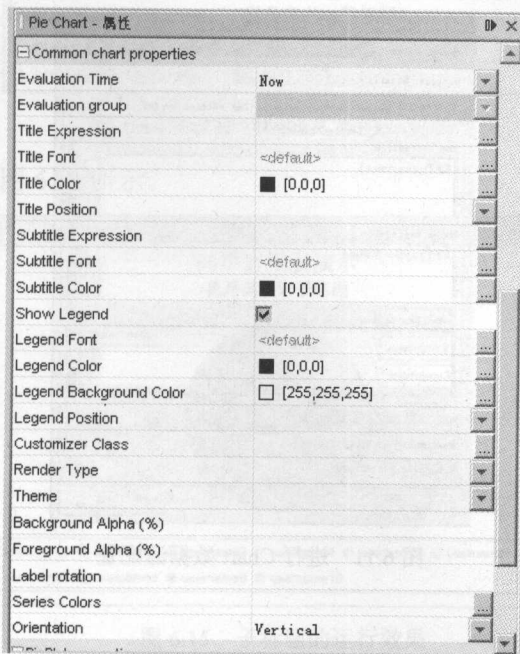


图 6.9 Chart 独有的属性

在图 6.9 中可以看到 Chart 控件的属性比较多, 但有一些已是前面学习过的知识, 这里不再赘述。

1. Evaluation Time 属性

Chart 中的数据可以进行配置, 即选择 Chart 控件, 然后单击鼠标右键, 在弹出的快捷菜单中选择 Chart Data 命令, 如图 6.10 所示, 弹出如图 6.11 所示的对话框, 可进行 Chart 数据源的配置。

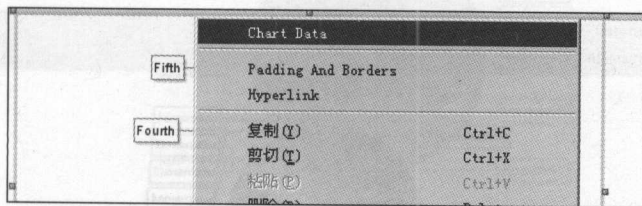


图 6.10 选择 Chart Data 命令

在图 6.11 中配置 $\$F\{username\}$ 和 $\$F\{buyBookNum\}$, 在什么时间执行这个表达式由 Evaluation Time 属性决定, 它的使用方法在前面章节已介绍过, 这里不再赘述。

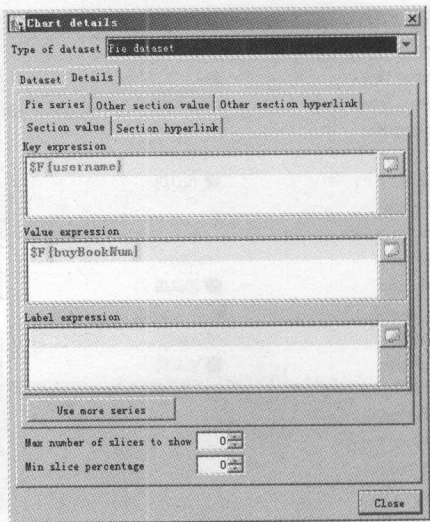


图 6.11 进行 Chart 数据源配置

2. 标题字体属性

设置 Chart 图表的标题属性，例如标题 Title 字体的配置如图 6.12 所示。

Title Expression	“高洪岩报表演示”
Title Font	宋体 18
Title Color	■ [255,0,0]
Title Position	Top

图 6.12 标题 Title 字体配置

属性 Title Position 决定了标题的位置，例如将 Title Position 设置为 Top，表示在图表的上方显示 Title 标题文本。

运行报表，标题字体的演示效果如图 6.13 所示。

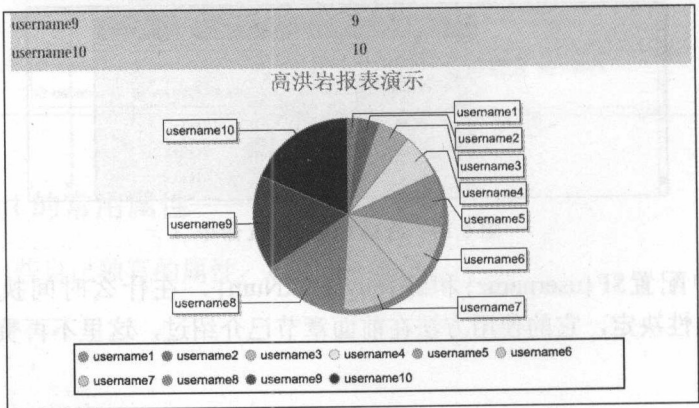


图 6.13 标题字体演示

3. 子标题字体属性

还可以设置子标题字体，例如子标题文本的外观属性设置如图 6.14 所示。

Subtitle Expression	"我是高洪岩子标题"
Subtitle Font	宋体 14
Subtitle Color	■ [0,0,255]

图 6.14 设置子标题文本的外观属性

子标题运行效果如图 6.15 所示。

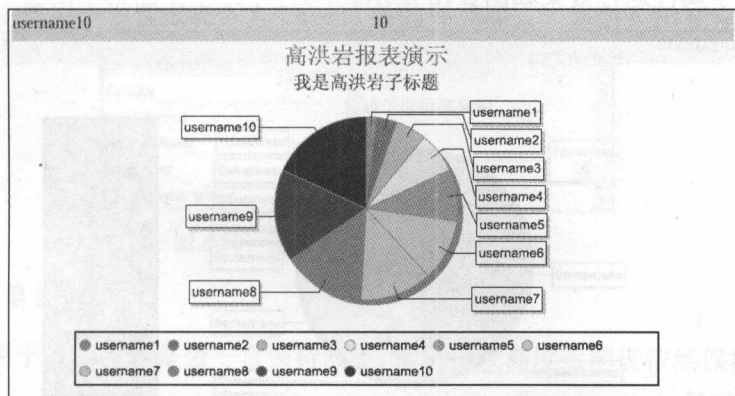


图 6.15 子标题的运行效果

4. Show Legend 属性

该属性的主要作用为是否显示下方的说明，若勾选复选框，则子标题的运行效果如图 6.15 所示，若不勾选该复选框，则子标题的运行效果如图 6.16 所示。

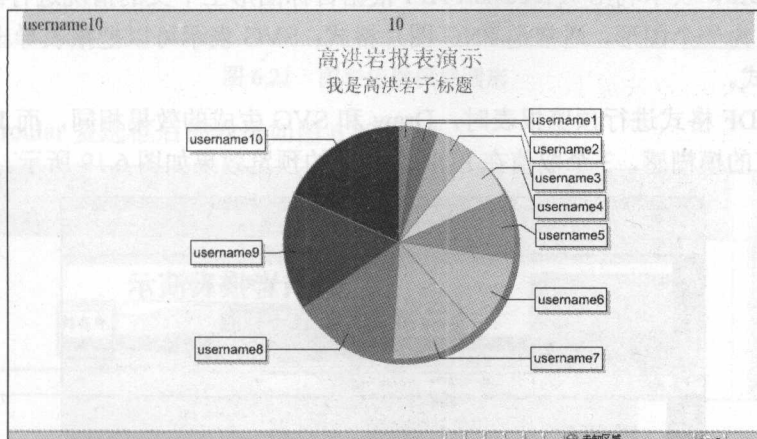


图 6.16 不勾选 Show Legend 属性的运行效果

5. Legend 的文字属性

勾选 Show Legend 属性后将显示一些选项，还可以对这些选项进行配置，Legend 的文字属性设置如图 6.17 所示。

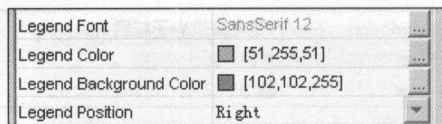


图 6.17 设置 Legend 的文字属性

Legend 的文字属性运行效果如图 6.18 所示。

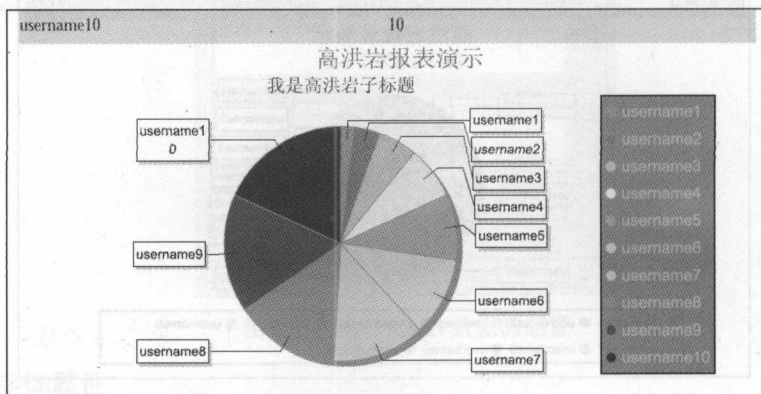


图 6.18 设置 Legend 的文字属性运行效果

6. Render Type 属性

该属性的主要作用是生成 Chart 图表图形文件的渲染类型，可以取 3 个值：Draw、Image 和 SVG，其中 Draw 表示利用 JFreeChart API 根据目标图形上下文的情况进行绘制；Image 表示将把报表导出为一个图形，通常是 PNG 图片格式；SVG 表示可以把报表导出为一个可缩放的图形 SVG 格式。

但在利用 PDF 格式进行预览报表时，Draw 和 SVG 生成的效果相同，而 Image 则会出现位图放大后特有的模糊感，3 个取值在 PDF 文件中的预览效果如图 6.19 所示。

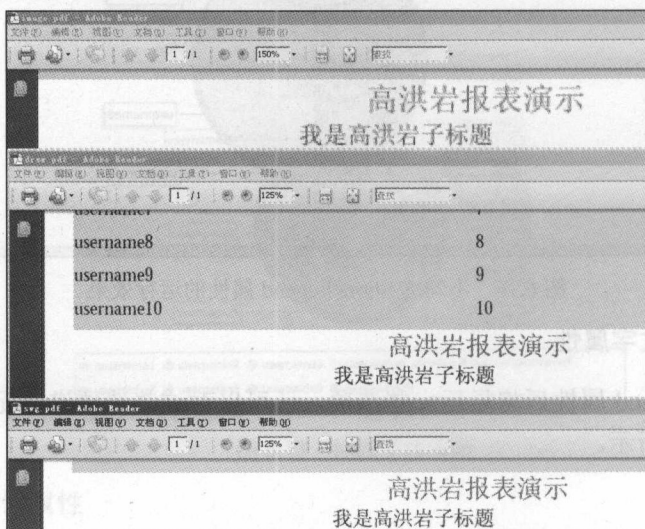


图 6.19 3 个值在 PDF 文件中的预览效果

另外这 3 个值在不同的预览格式中的效果也不相同，如预览方式使用 HTML，则用 SVG 值生成的图片就会带有锯齿状，这要根据不同的预览格式来进行选择，如果是 PDF 格式的，建议还是使用 Draw 值。

7. PiePlot 属性

PiePlot 属性主要用于设置报表下方 Legend 条的外观样式，如图 6.20 所示。

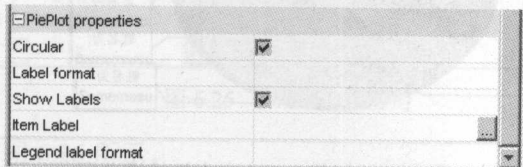


图 6.20 报表下方 Legend 条的属性

(1) Circular 属性

该属性主要用于决定图表是否一直保持圆形，缩小报表高度后图表依然保持圆形的效果如图 6.21 所示。

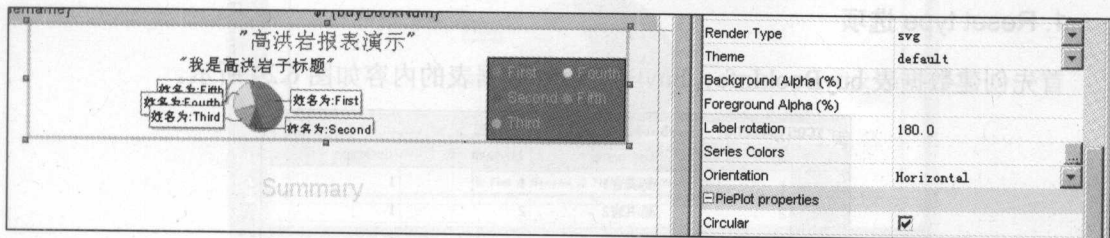


图 6.21 图表依然保持圆形

取消勾选 Circular 复选框后的效果如图 6.22 所示。

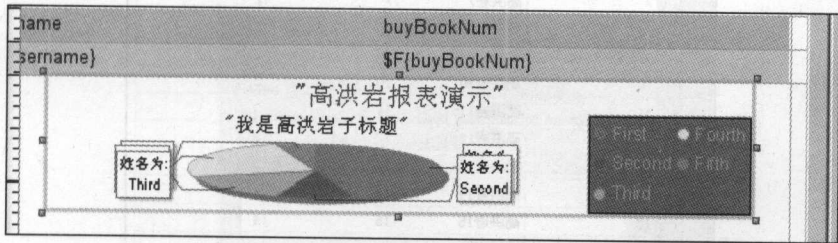


图 6.22 图表变形

(2) Label format 属性

该属性用于决定 Label 标题是否显示 format 格式，例如设置值为“姓名:{0}”，运行效果如图 6.23 所示。

(3) 其他属性

属性 Show Labels 用于决定是否显示 Label 标签，属性 Item Label 用于设置标签的字体及颜色，属性 Legend label format 用于决定 Legend 里文字的 format 字符。

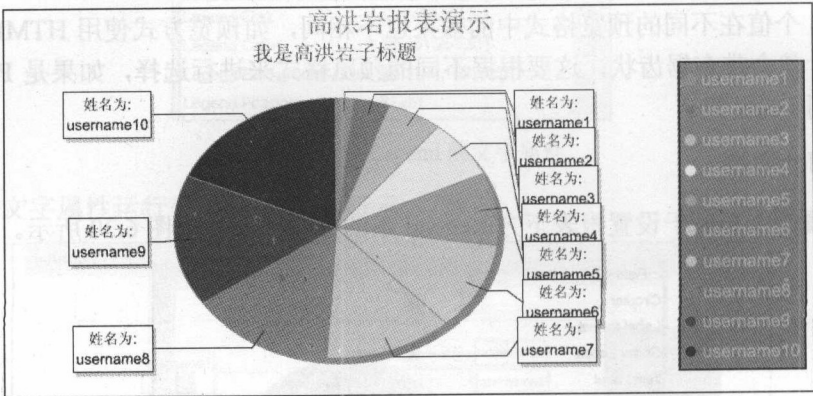


图 6.23 标签 Label 文本被格式化了

6.1.6 图表 Chart 的常用选项——饼状图

图表 Chart 中有很多选项可以设置，这些选项对能否正确使用 Chart 图表将起到至关重要的作用，下面将为大家进行介绍。

1. Reset type 选项

首先创建数据表 buyBookInfo，buyBookInfo 数据表的内容如图 6.24 所示。

TC05\SQL200... buyBookInfo				
	id	username	buyNum	booktype
	1	高洪岩1	1	1
	2	高洪岩2	2	1
	3	高洪岩3	3	1
	4	高洪岩4	4	1
	5	高洪岩5	5	1
	6	高洪岩6	6	2
	7	高洪岩7	7	2
	8	高洪岩8	8	2
	9	高洪岩9	9	2
	10	高洪岩10	10	3
	11	高洪岩11	11	3
	12	高洪岩12	12	3
	13	高洪岩13	13	3
	14	高洪岩14	14	4
	15	高洪岩15	15	4
	16	高洪岩16	16	4
	17	高洪岩17	17	4
	18	高洪岩18	18	5
	19	高洪岩19	19	5
	20	高洪岩20	20	6
*	NULL	NULL	NULL	NULL

图 6.24 buyBookInfo 数据表的内容

设置数据源并添加 Fields 对象，添加 SQL 语句如图 6.25 所示。

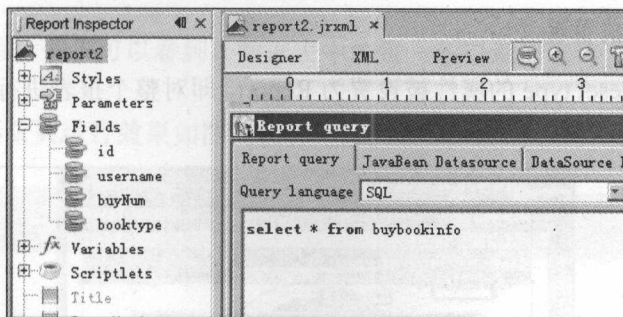


图 6.25 添加 SQL 语句

设计报表模板，如图 6.26 所示。

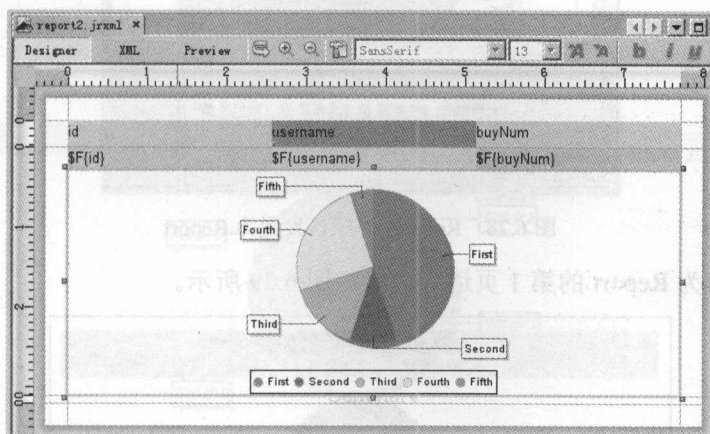


图 6.26 设计报表模板

设置 Chart 控件的数据源属性，如图 6.27 所示。

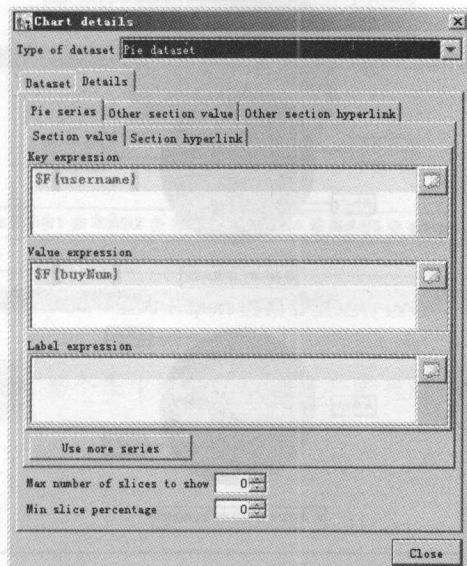


图 6.27 设计数据源属性

(1) 值为 Report 的情况

在默认情况下，Reset type 的属性被设置为 Report，即对整个报表进行统计显示，如图 6.28 所示。

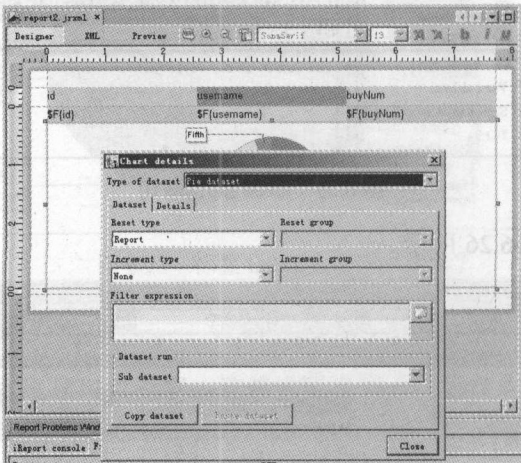


图 6.28 Reset type 默认设置为 Report

运行报表，值为 Report 的第 1 页运行效果如图 6.29 所示。

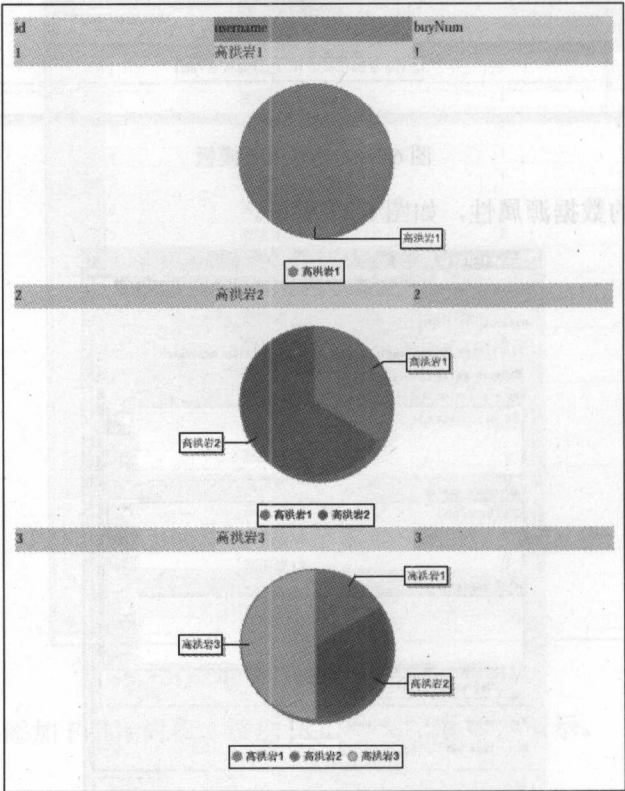


图 6.29 值为 Report 的第 1 页运行效果

从图 6.29 中的运行效果可以看到, 数据表中的每一条记录都参与到图表的展示中, 这就是 Report 值的作用。

值为 Report 的第 2 页运行效果如图 6.30 所示。

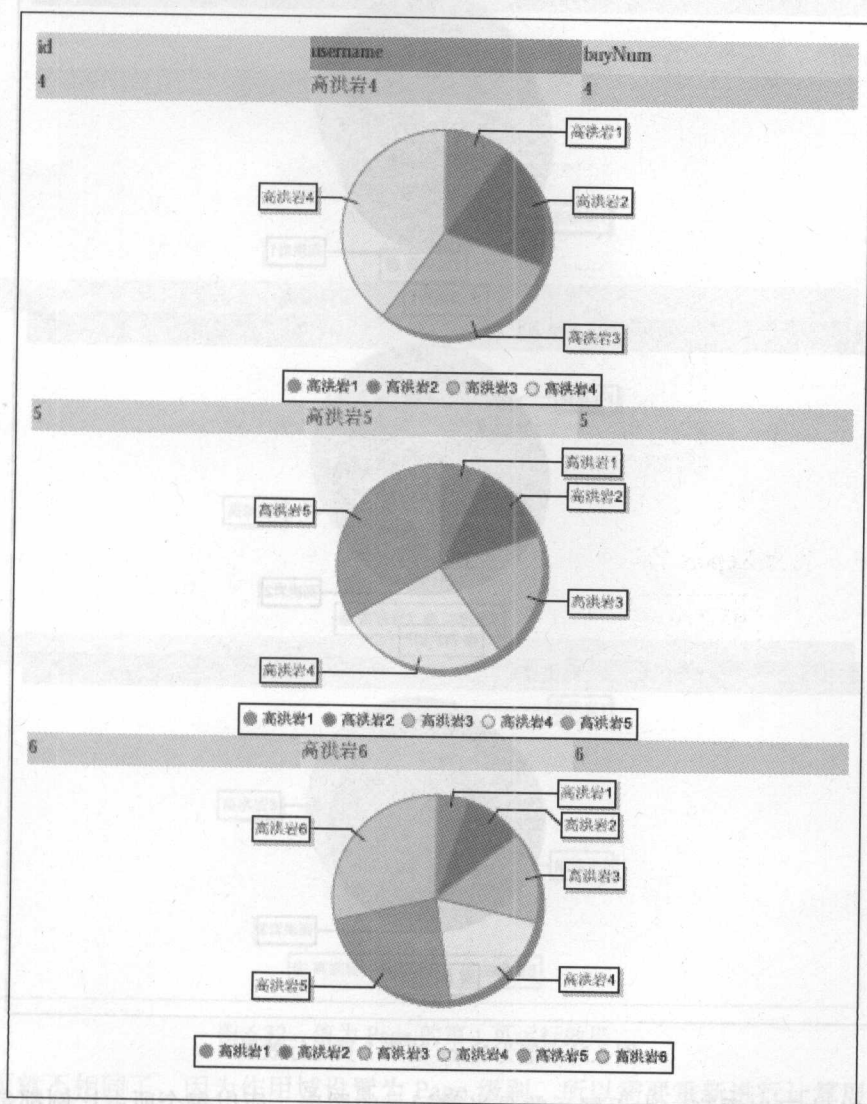


图 6.30 值为 Report 的第 2 页运行效果



提示

选择 Report 就是要对整个表中的所有数据进行图表统计显示。

(2) 值为 None 的情况

如果设置为 None, 则表示只把当前的记录作为图表展示的内容, 运行报表, 值为 None

的第 1 页运行效果如图 6.31 所示。

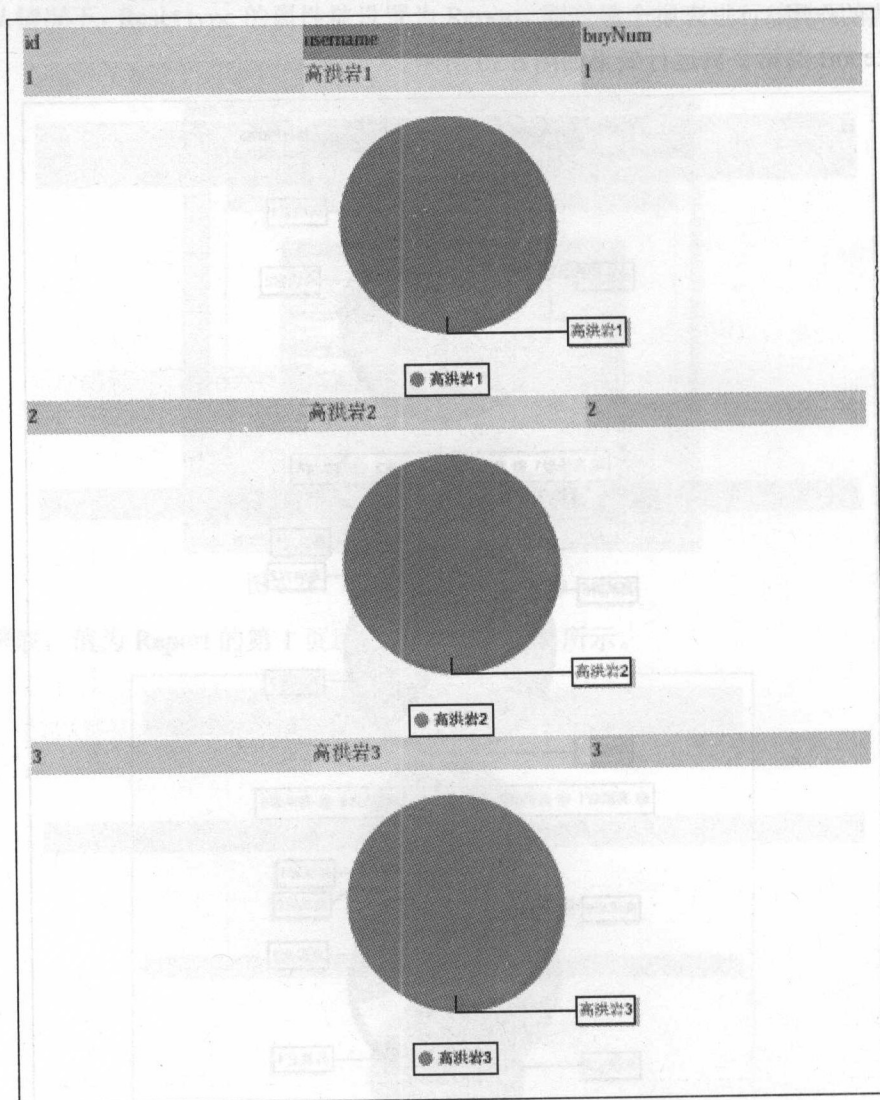


图 6.31 值为 None 的第 1 页运行效果

从图 6.31 中可以看到，由于展示的是当前记录的图表，所以每个所占比例都是 100%。

(3) 值为 Page 的情况

设置为 Page 的第 1 页运行效果与值为 Report 的第 1 页运行效果相同，如图 6.32 所示。

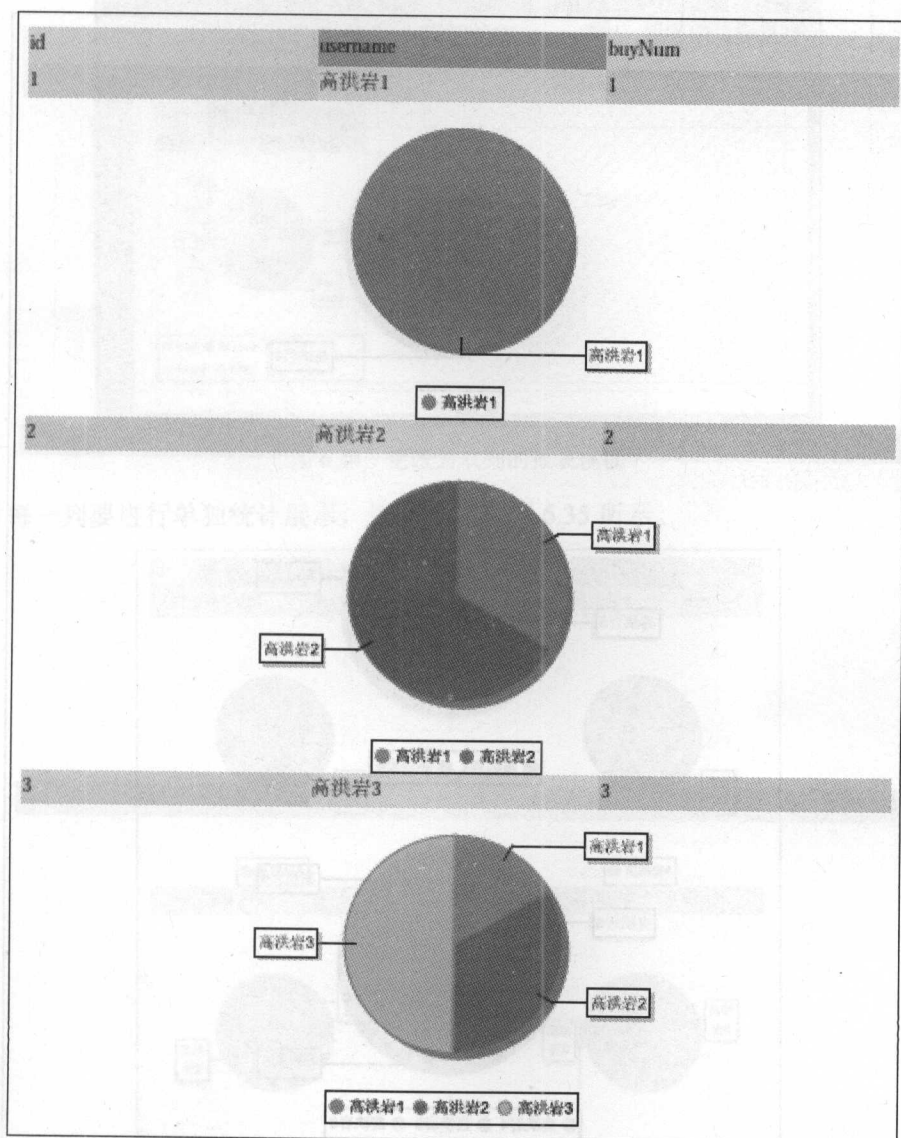


图 6.32 值为 Page 的第 1 页运行效果

但第 2 页就不相同了，因为作用域设置为 Page 级别，所以需要重新进行计算展示，如图 6.33 所示。

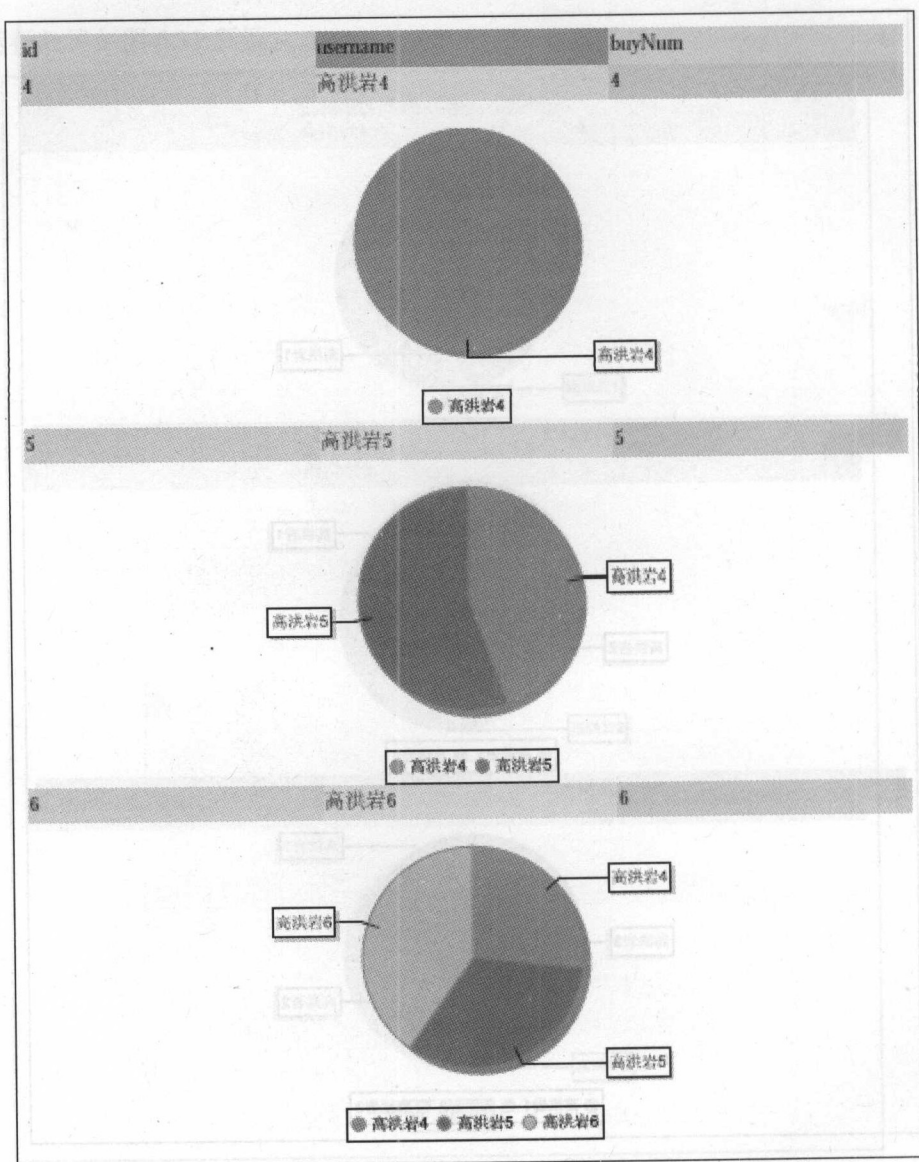


图 6.33 值为 Page 的第 2 页运行效果

(4) 值为 Column 的情况

设置为 Column 后，即更改为双列的报表模板，报表外观如图 6.34 所示。

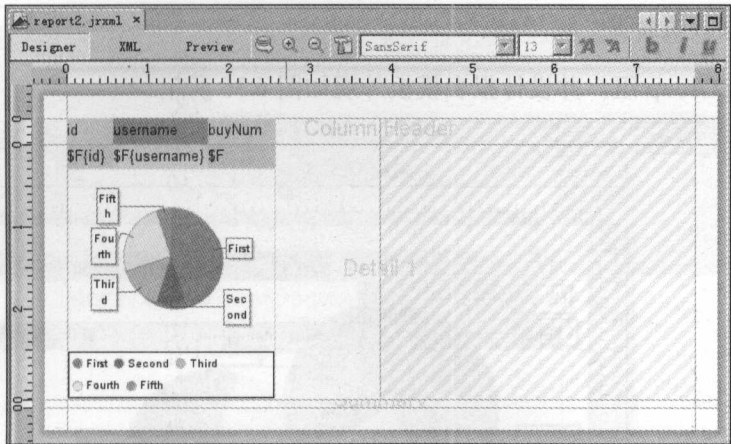


图 6.34 更改为双列的报表模板

这时每一列要进行单独统计展示，运行效果如图 6.35 所示。

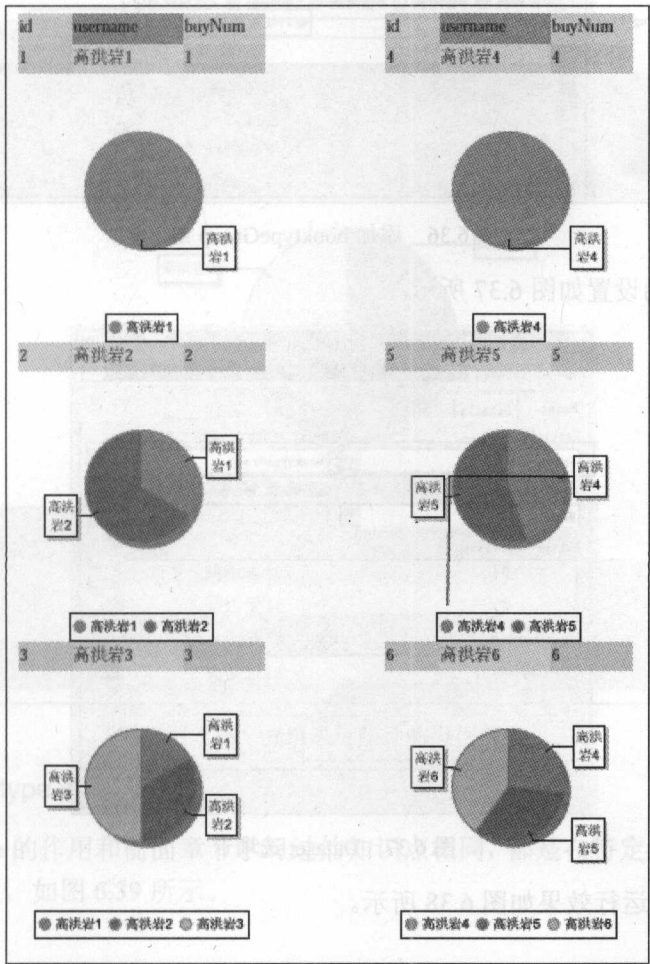


图 6.35 每一列进行单独统计展示

(5) 值为 Group 的情况

添加 booktypeGroup 组，并设计报表模板外观如图 6.36 所示。

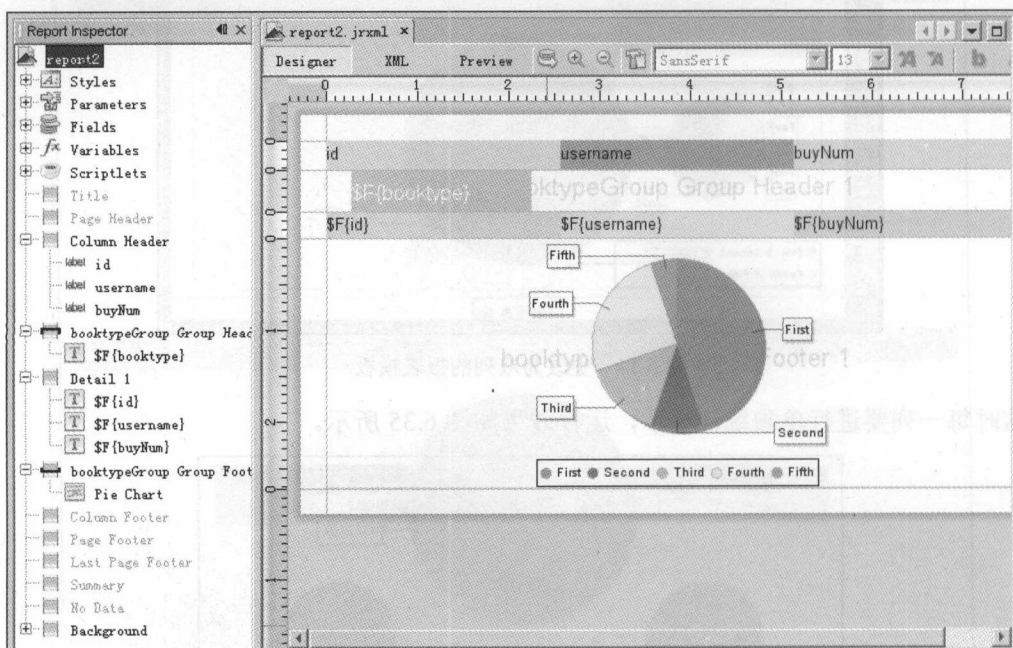


图 6.36 添加 booktypeGroup 组

Dataset 选项卡的设置如图 6.37 所示。

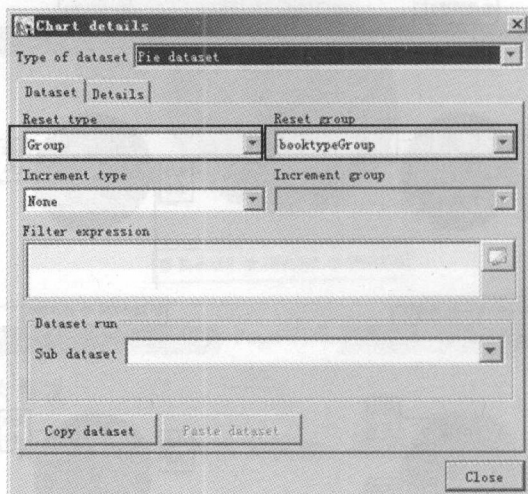


图 6.37 Dataset 选项卡

分组进行统计的运行效果如图 6.38 所示。

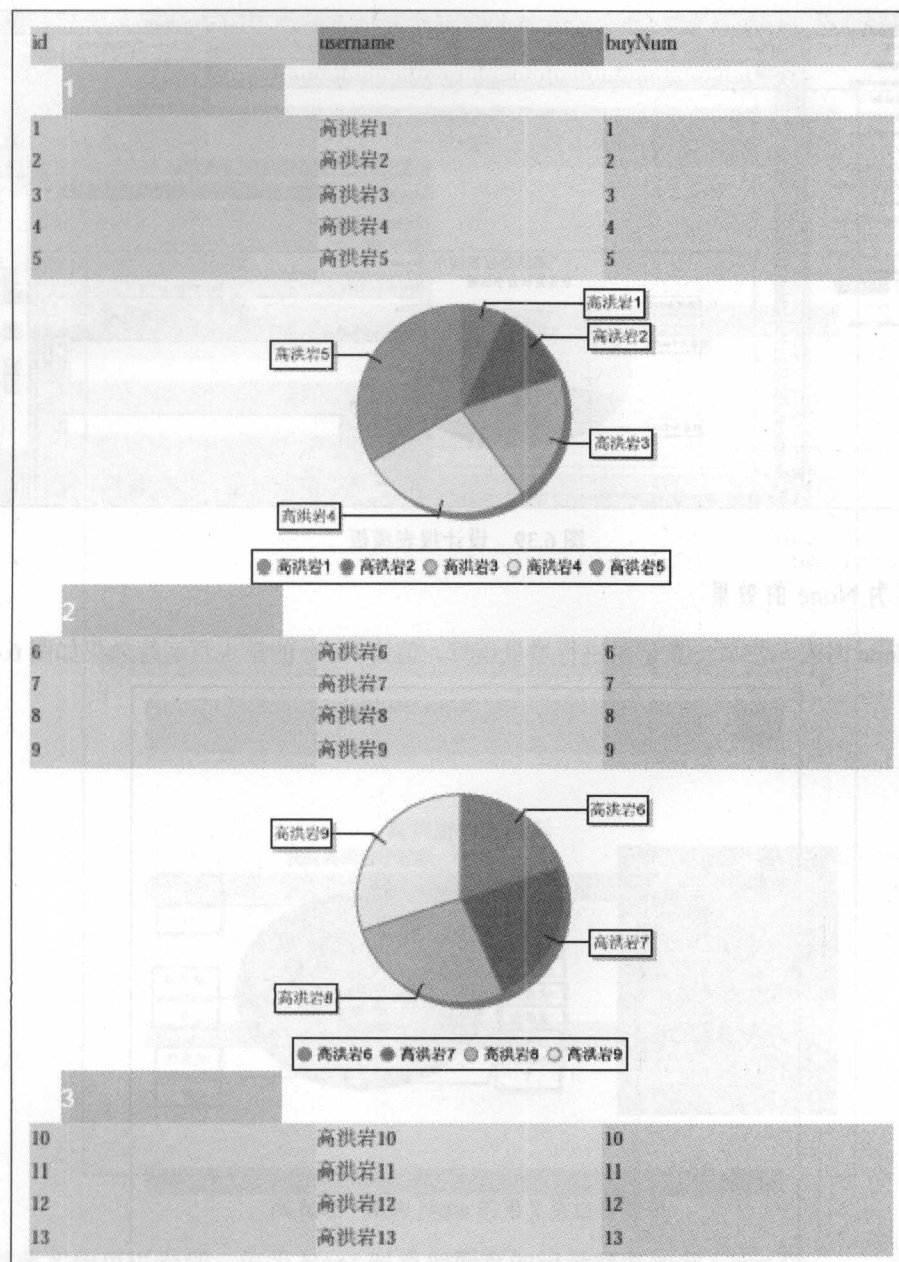


图 6.38 分组进行统计的运行效果

2. Increment type 选项

Increment type 的作用和前面章节学习过的知识点相同，都是在特定情况下进行自增运算。设计报表模板，如图 6.39 所示。

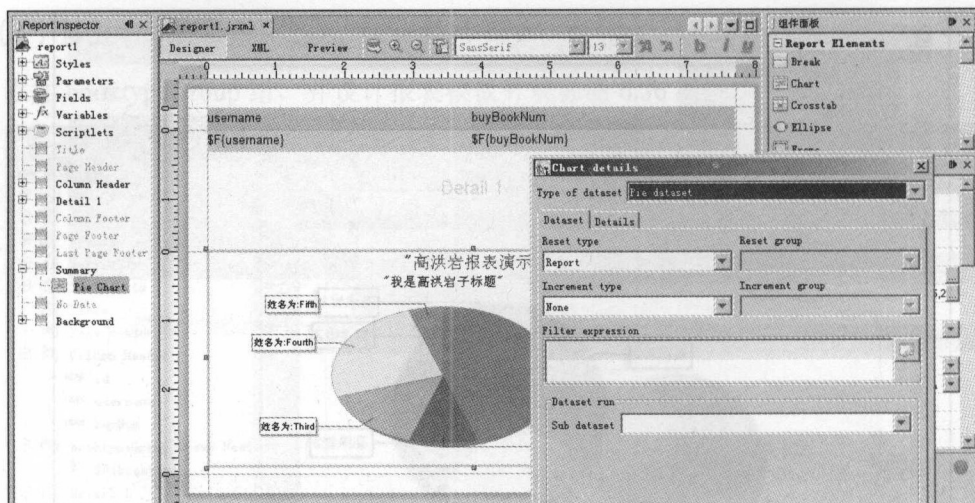


图 6.39 设计报表模板

(1) 值为 None 的效果

值为 None 时表示将每一条记录进行增量运算, 值为 None 的第 1 页运行效果如图 6.40 所示。

username	buyBookNum
username1	1
username2	2
username3	3
username4	4
username5	5
username6	6

图 6.40 值为 None 的第 1 页运行效果

值为 None 的第 2 页运行效果如图 6.41 所示。

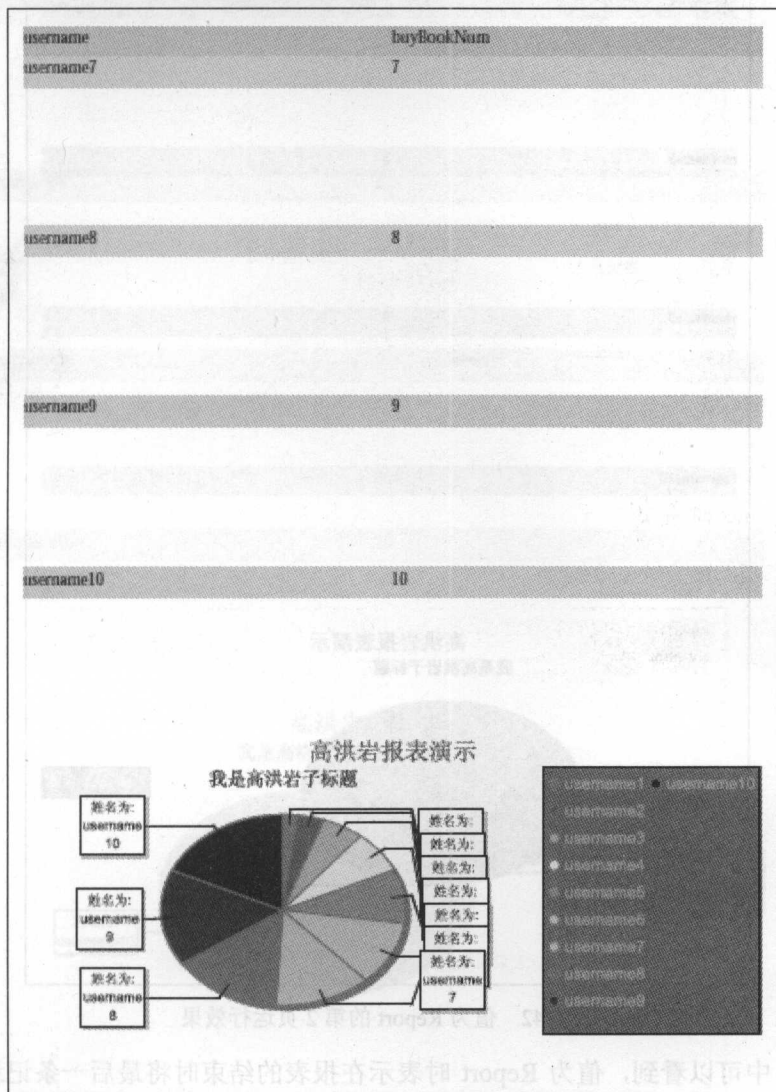


图 6.41 值为 None 的第 2 页运行效果

通过此实验可以看到，值为 None 时将把所有的记录都添加到 Chart 中。

(2) 值为 Report 的效果

值为 Report 的第 1 页运行效果和值为 None 的第 1 页运行效果相同，但第 2 页效果则不相同，值为 Report 的第 2 页运行效果如图 6.42 所示。

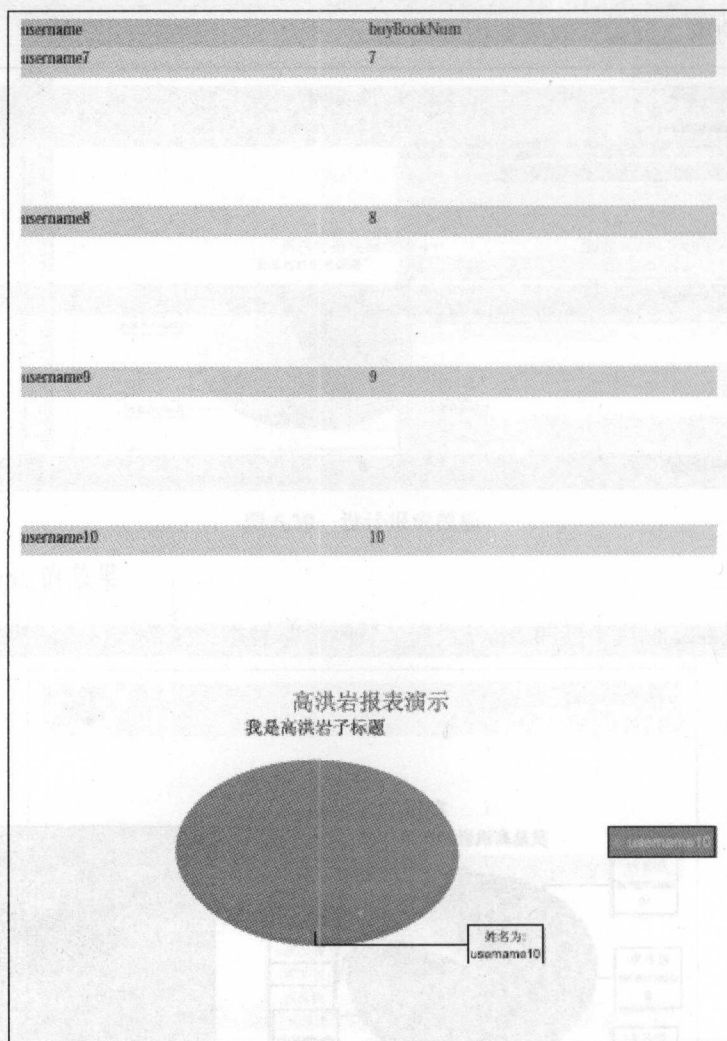


图 6.42 值为 Report 的第 2 页运行效果

从图 6.42 中可以看到，值为 Report 时表示在报表的结束时将最后一条记录填充进 Chart 图表中。

(3) 值为 Page 的效果

值为 Page 的第 1 页运行效果和值为 None 的第 1 页运行效果相同，但第 2 页效果不相同，值为 Page 的第 2 页运行效果如图 6.43 所示。

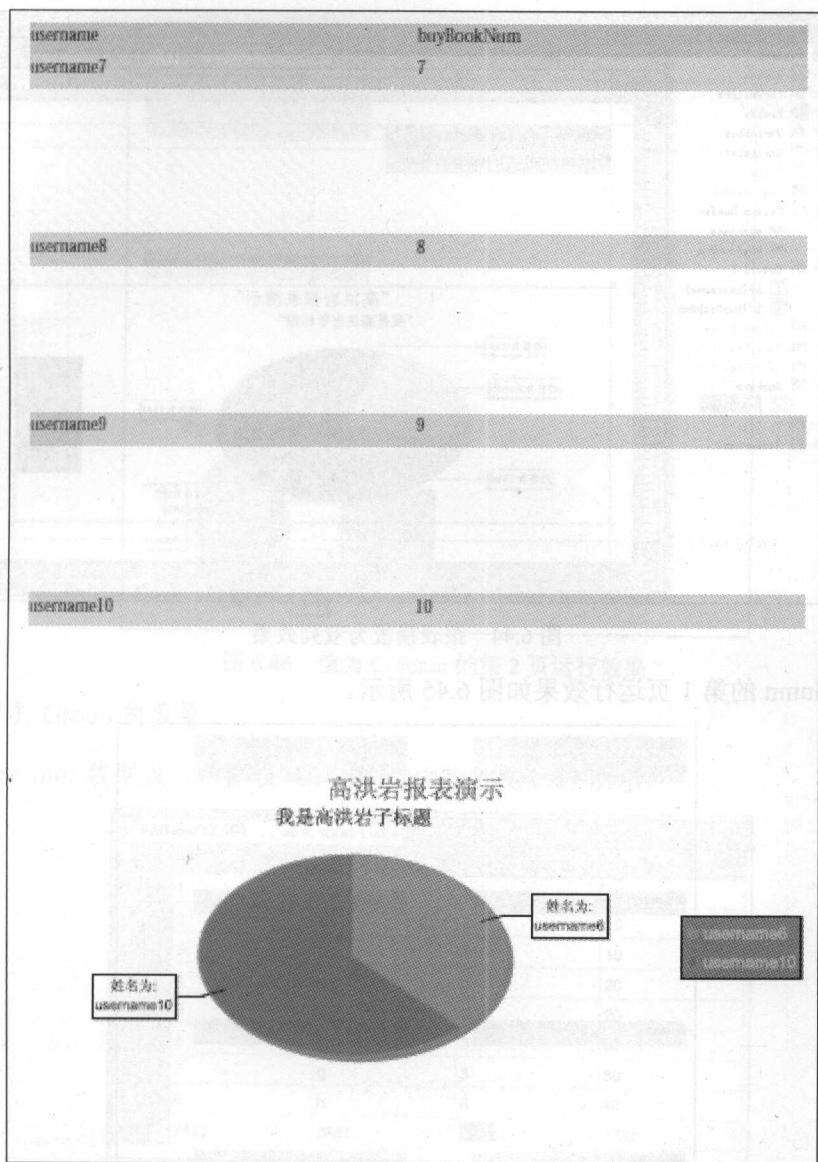


图 6.43 值为 Page 的第 2 页运行效果

值为 Page 表示将每一页的最后一条记录添加到 Chart 图表中进行运算演示。

(4) 值为 Column 的效果

值为 Column 表示设计报表模板为双列，效果如图 6.44 所示。

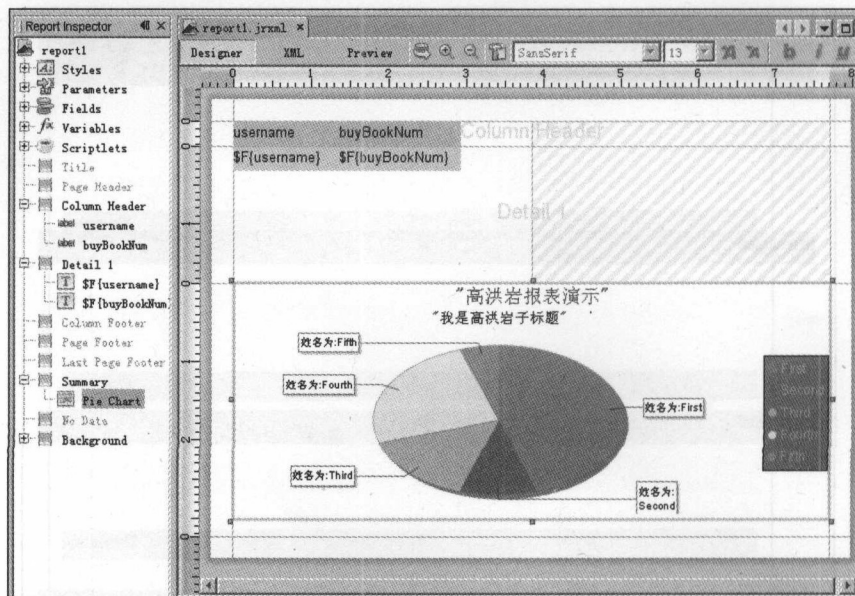


图 6.44 报表模板为双列效果

值为 Column 的第 1 页运行效果如图 6.45 所示。

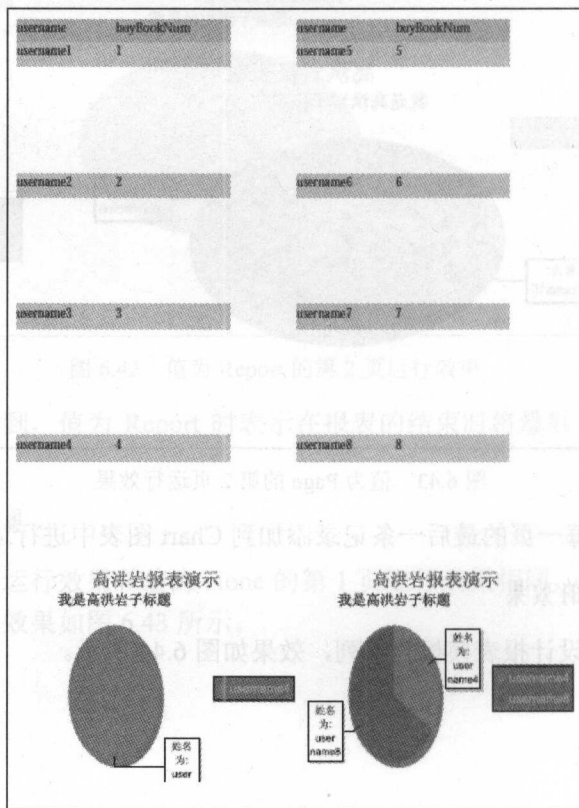


图 6.45 值为 Column 的第 1 页运行效果

值为 Column 的第 2 页运行效果如图 6.46 所示。

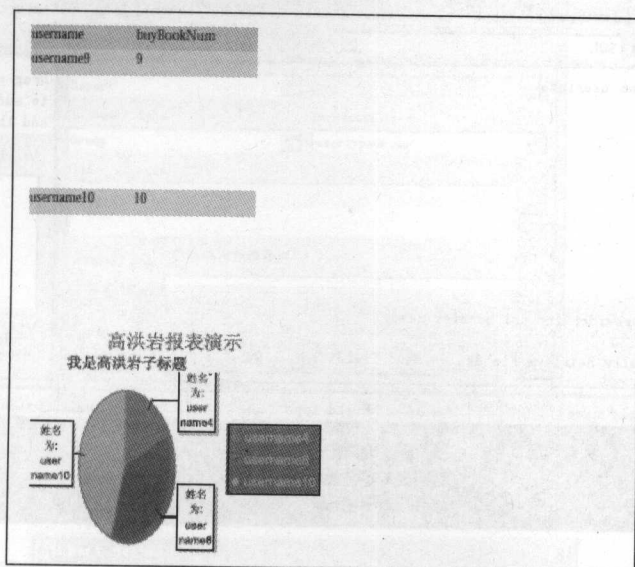


图 6.46 值为 Column 的第 2 页运行效果

(5) 值为 Group 的效果

设计 userinfo 数据表，数据表 userinfo 的内容如图 6.47 所示。

BAOGAOTING\... dba. userinfo				
	id	username	age	usertype
	1	a	2	10
	2	b	2	10
	3	c	2	10
	4	d	3	20
	5	e	3	20
	6	f	3	30
	7	g	3	30
	8	h	6	40
**	NULL	NULL	NULL	NULL

图 6.47 设计数据表 userinfo 内容

利用 iReport 连接 userinfo 数据表，设置 JDBC 为数据源，如图 6.48 所示。

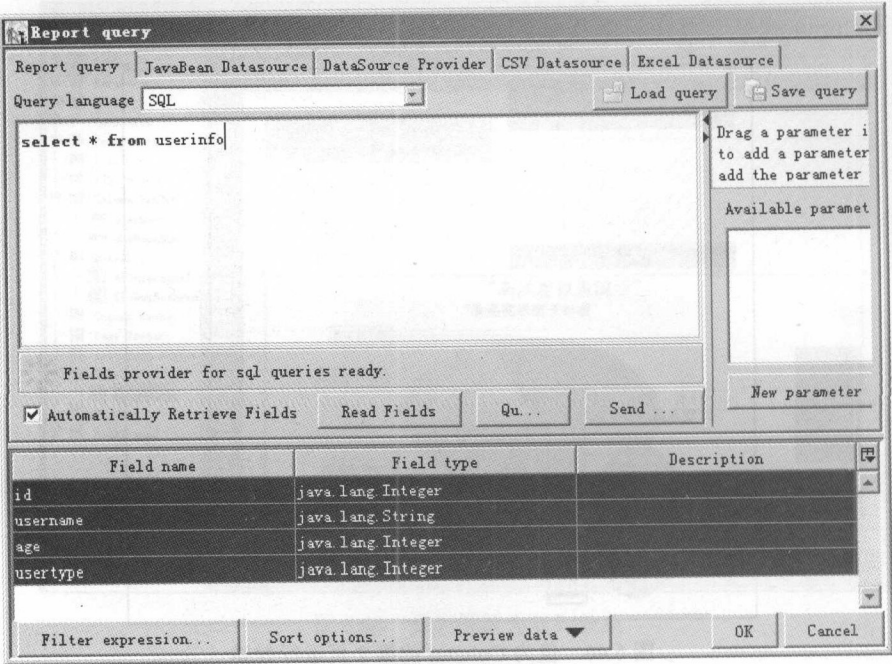


图 6.48 设置 JDBC 为数据源

重新设计报表模板，并添加一个 Variables 变量，如图 6.49 所示。

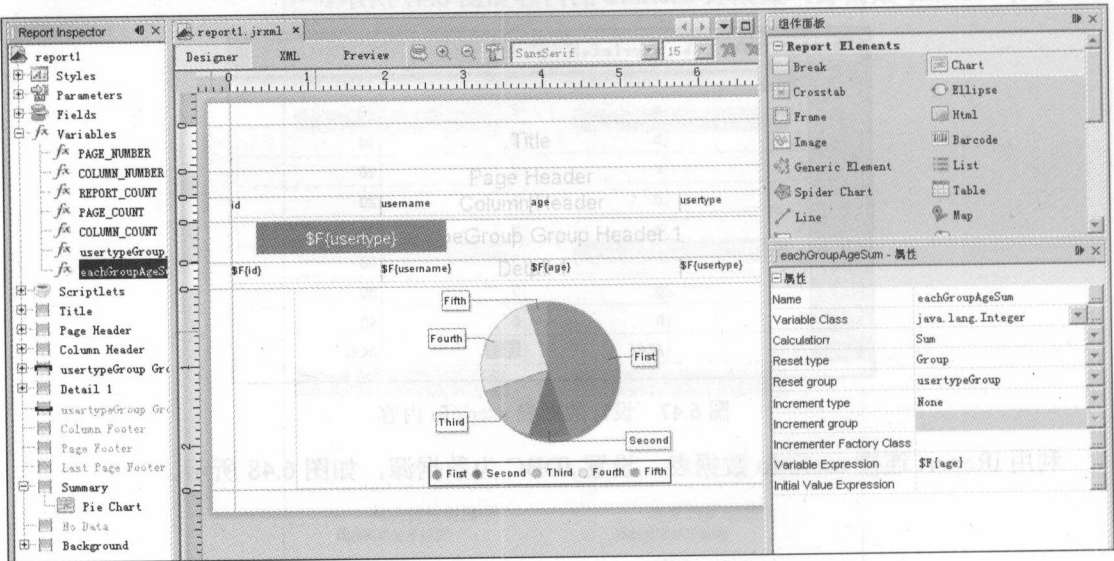


图 6.49 重新设计报表模板并添加一个 Variables 变量

设计 Chart 属性，Dataset 选项卡的设置如图 6.50 所示。

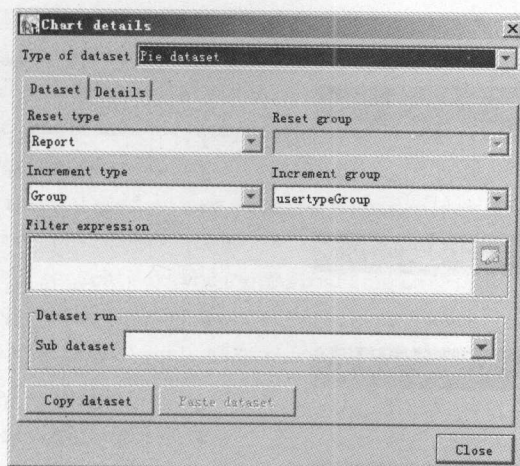


图 6.50 Dataest 选项卡

Details 选项卡的设置如图 6.51 所示。

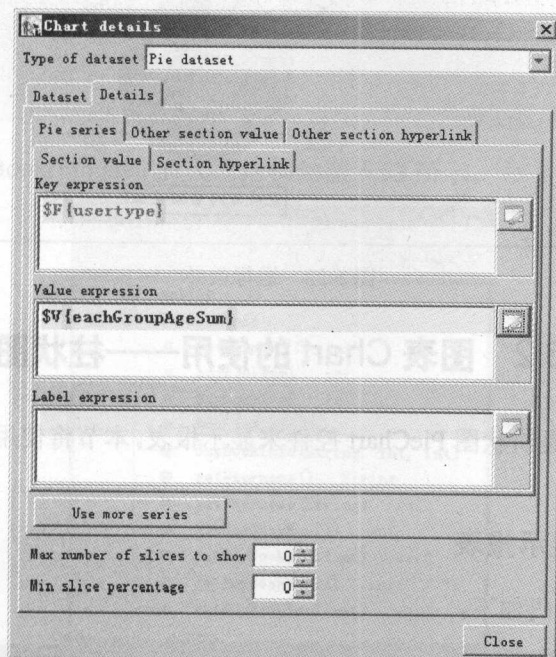


图 6.51 Details 选项卡

报表运行效果如图 6.52 所示。

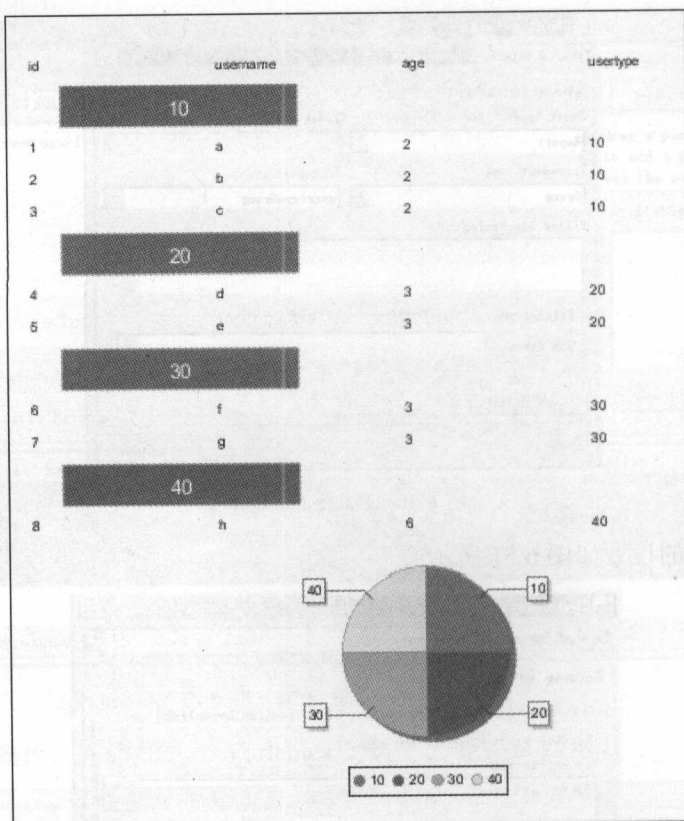


图 6.52 运行效果

6.2 图表 Chart 的使用——柱状图

前面的小节使用的是饼状图 PieChart 控件来显示报表,本节将使用全新的控件——柱状图来显示报表。

6.2.1 使用柱状图显示报表

新建 Servlet 核心代码如下:

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List buyBookInfoList = new ArrayList();
            for (int i = 0; i < 10; i++)
            {
                buyBookInfoList.add(new BuyBookInfo("username" + (i + 1),
                    (i + 1), (i + 2)));
            }
            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")

```

```

+ "jrxml\\report1.jrxml";
String jrxmlDestPathMain = this.getClass().getClassLoader()
    .getResource("").getPath().substring(1)+"jasperreports/report1.jasper";
JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
    jrxmlDestPathMain);
InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
ServletOutputStream sosRef = response.getOutputStream();
response.setContentType("application/pdf");
JasperRunManager.runReportToPdfStream(isRef, sosRef, new HashMap(),
    new JRBeanCollectionDataSource(buyBookInfoList));
sosRef.flush();
sosRef.close();
}
catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

新建实体类，BuyBookInfo.java 的代码结构如图 6.53 所示。

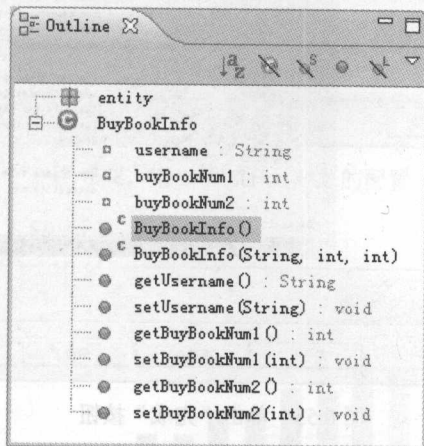


图 6.53 实体类结构

新建报表模板，添加 Chart 控件并选择柱状图，如图 6.54 所示。

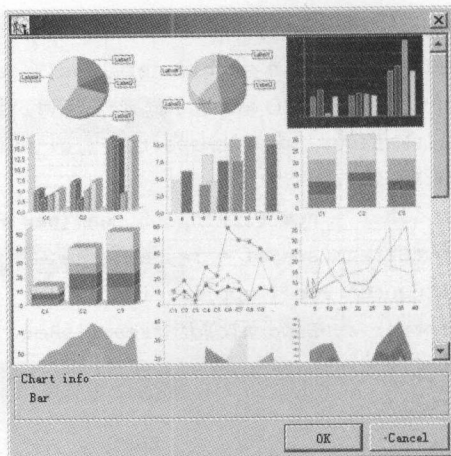


图 6.54 选择柱状图

单击 OK 按钮，弹出如图 6.55 所示的对话框，在该对话框中直接单击“完成”按钮完成配置。

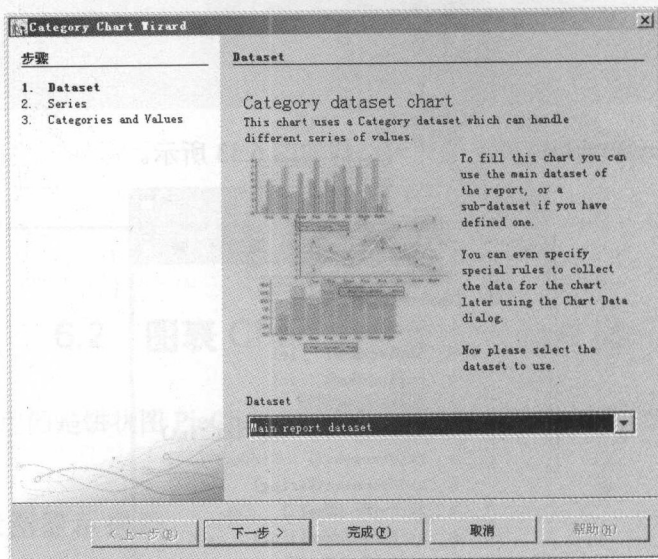


图 6.55 单击“完成”按钮

重新进入 Chart Data 菜单，添加两个 Series 对象，如图 6.56 所示。

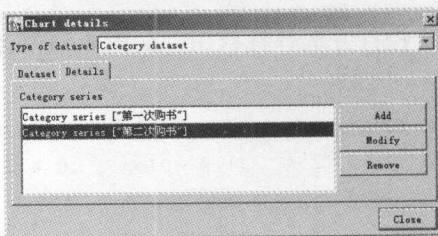


图 6.56 添加两个 Series 对象

其中第 1 个 Series 对象的配置如图 6.57 所示。

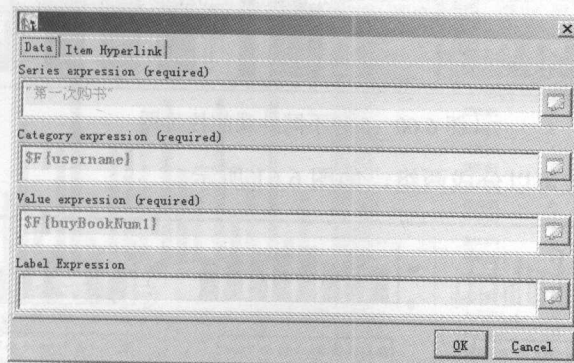


图 6.57 第 1 个 Series 对象的配置

第 2 个 Series 对象的配置如图 6.58 所示。

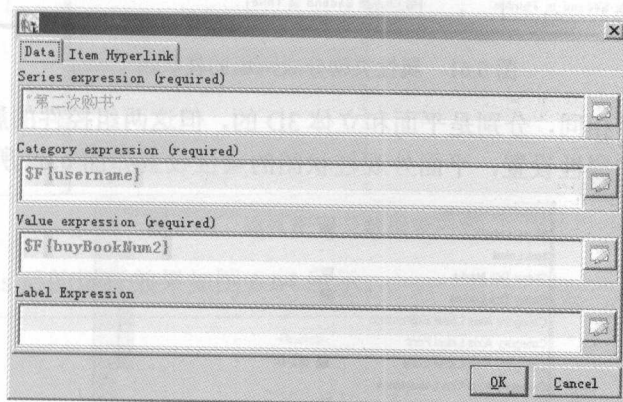


图 6.58 第 2 个 Series 对象的配置

程序运行后的效果如图 6.59 所示。

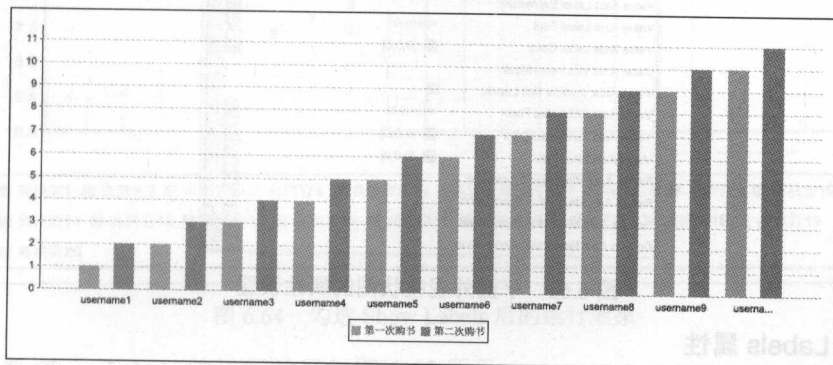


图 6.59 程序运行结果

6.2.2 图表 Chart 的常用属性——柱状图

在 iReport 中，柱状图一共有 5 种外观样式，如图 6.60 所示。

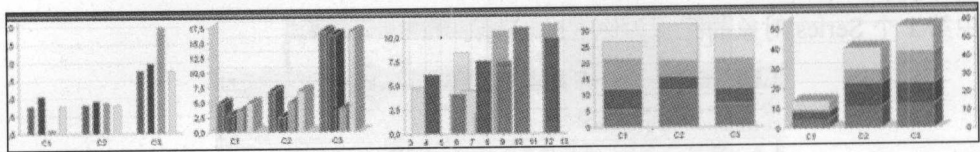


图 6.60 5 种不同外观的柱状图

这 5 个柱状图的属性可以分成两组，如图 6.61 所示。

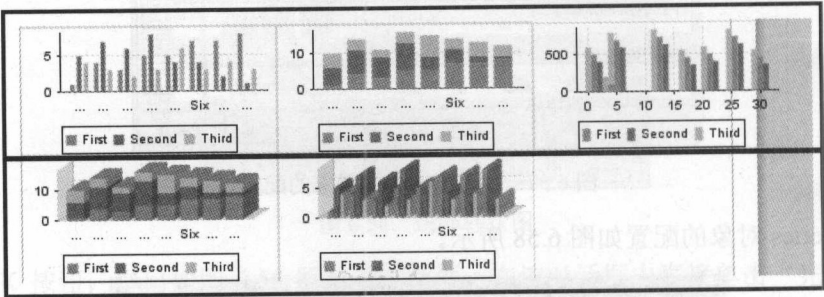


图 6.61 属性类别分成两组的柱状图

这两组的外观不尽相同，分别是平面和立体 3D 的，但这两组控件的属性大体相同。先来学习一下前 3 个控件的属性设置，平面外观柱状图的属性设置如图 6.62 所示。

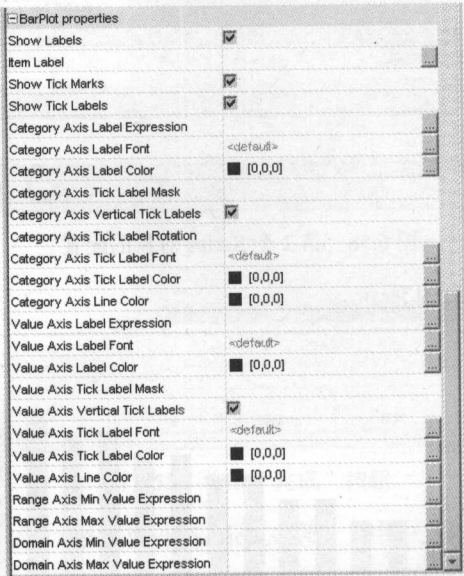


图 6.62 平面外观柱状图的属性设置

1. Show Labels 属性

报表模板的属性配置及数据表内容如图 6.63 所示。

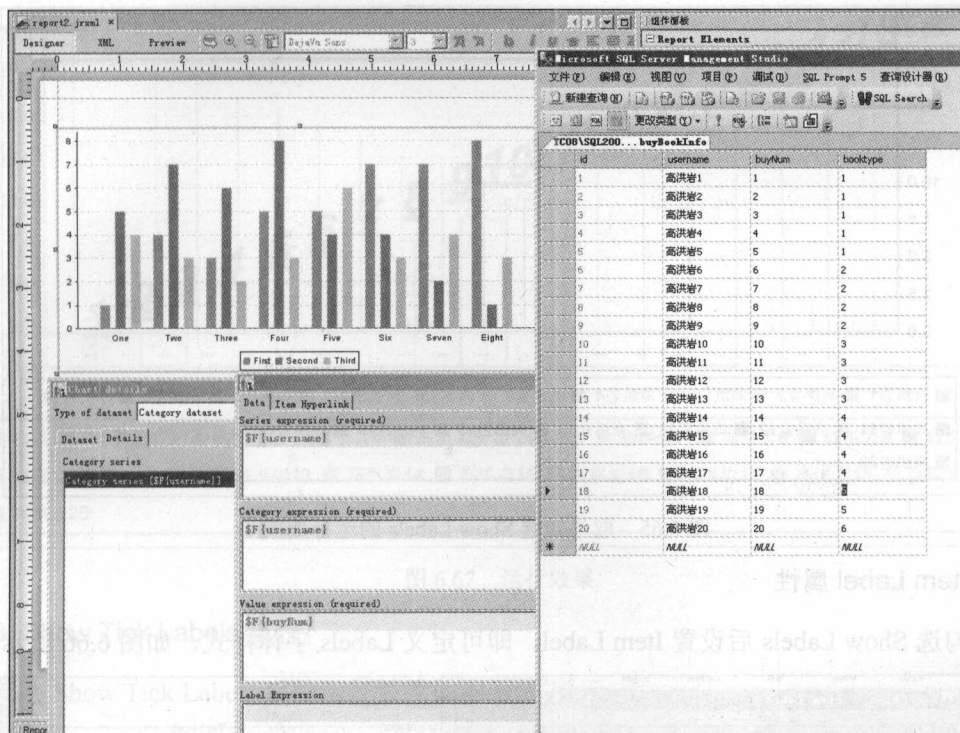


图 6.63 属性配置及数据表内容

勾选 Show Labels 后的运行效果如图 6.64 所示。

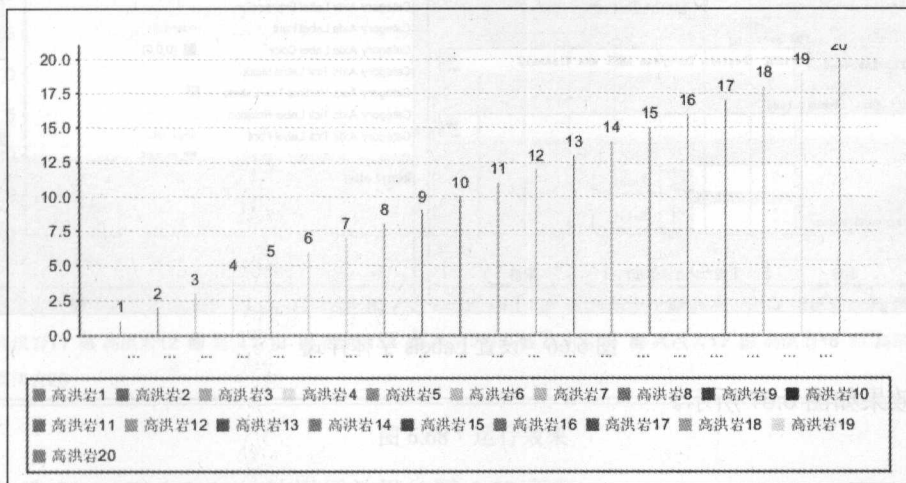


图 6.64 勾选 Show Labels 后的运行效果

取消勾选 Show Labels 的运行效果如图 6.65 所示。

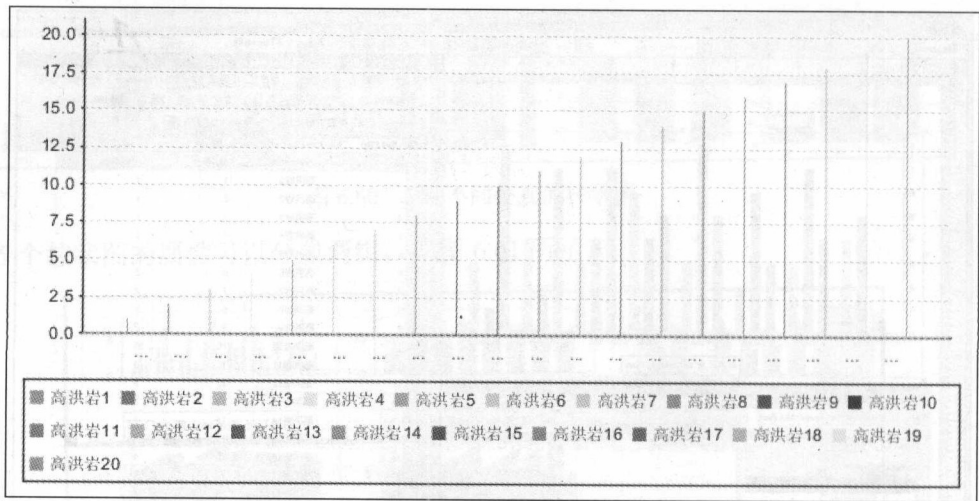


图 6.65 取消勾选 Show Labels 的运行效果

2. Item Label 属性

在勾选 Show Labels 后设置 Item Label，即可定义 Labels 字体样式，如图 6.66 所示。

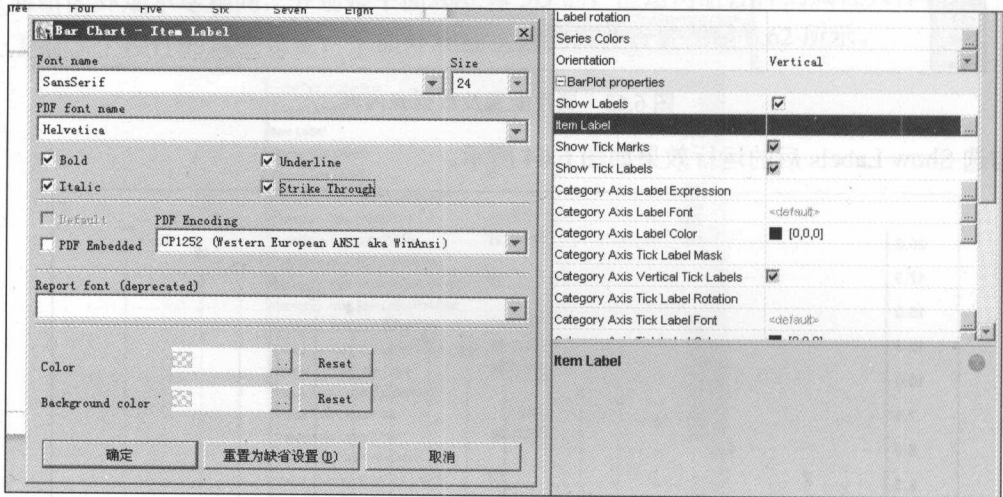


图 6.66 设置 Labels 字体样式

运行效果如图 6.67 所示。

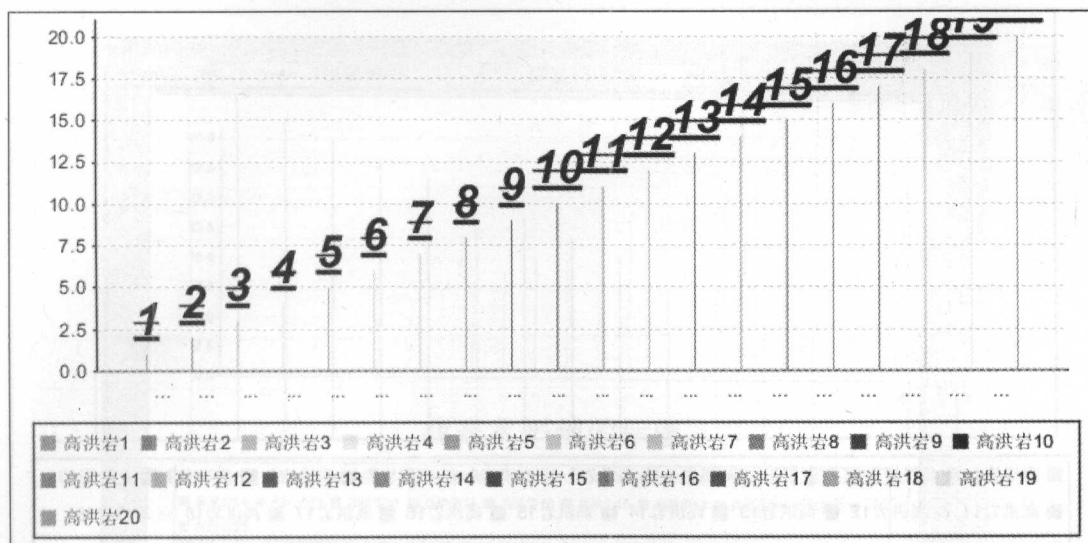


图 6.67 运行效果

3. Show Tick Labels 属性

勾选 Show Tick Labels 后的运行效果如图 6.68 所示。

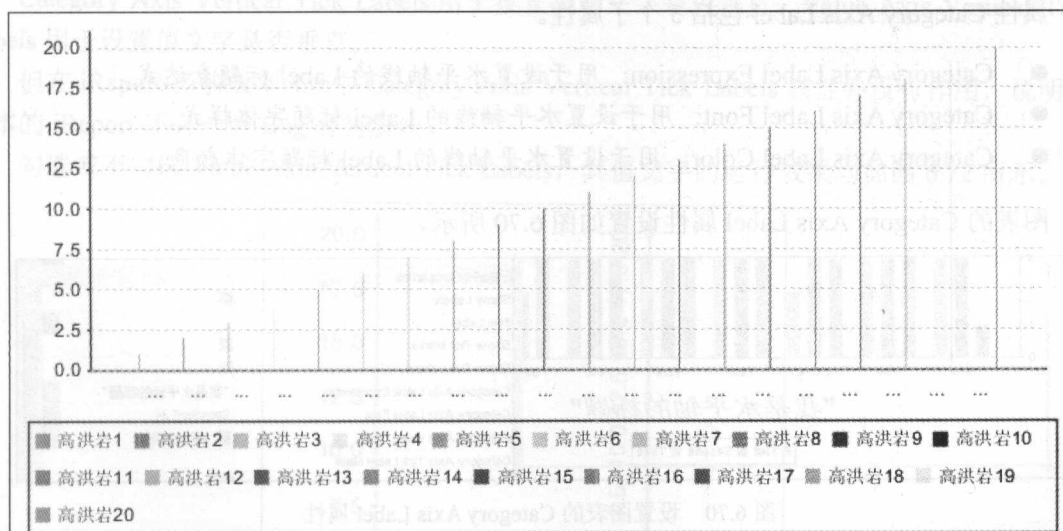


图 6.68 运行效果

未勾选 Show Tick Labels 的运行效果如图 6.69 所示。

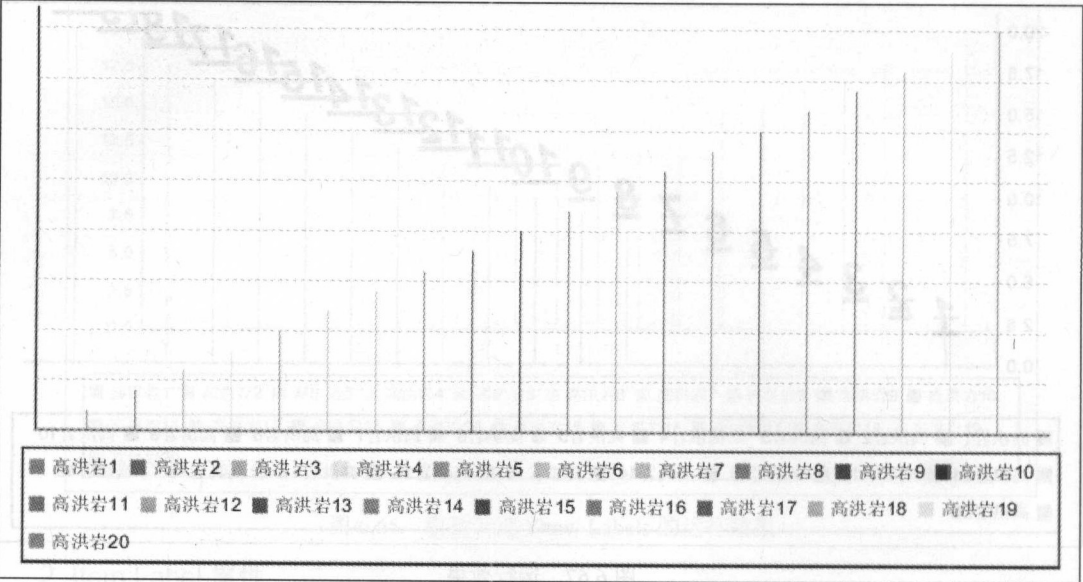


图 6.69 未勾选的运行效果

4. Category Axis Label 属性

属性 Category Axis Label 包括 3 个子属性。

- Category Axis Label Expression: 用于设置水平轴线的 Label 标题表达式。
- Category Axis Label Font: 用于设置水平轴线的 Label 标题字体样式。
- Category Axis Label Color: 用于设置水平轴线的 Label 标题字体颜色。

图表的 Category Axis Label 属性设置如图 6.70 所示。



图 6.70 设置图表的 Category Axis Label 属性

运行效果如图 6.71 所示。

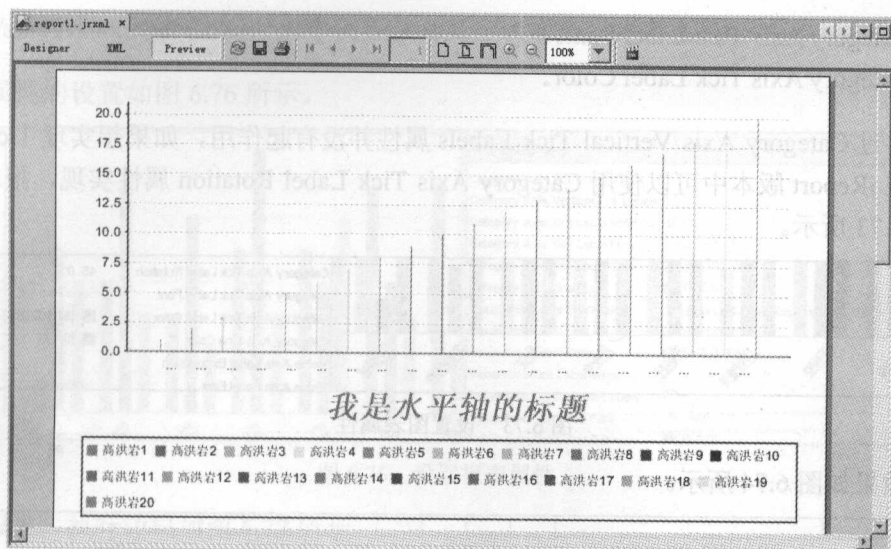


图 6.71 运行效果

5. Category/Value Axis Vertical Tick Labels 属性

Category Axis Vertical Tick Labels 用于设置类别文字是否垂直。Value Axis Vertical Tick Labels 用于设置值文字是否垂直。

但在 iReport 当前版本 4.6 中 Category Axis Vertical Tick Labels 属性并没有作用，说明该版本的 iReport 还有一些 Bug 有待解决。

勾选或不勾选 Value Axis Vertical Tick Labels，其值文字的运行效果均如图 6.72 所示。

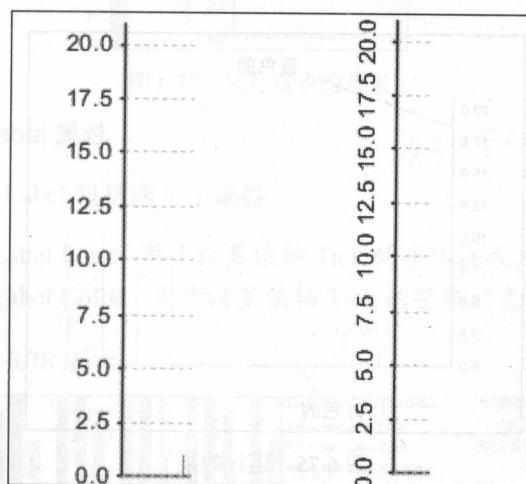


图 6.72 运行效果

6. Category Axis Tick Label 属性

属性 Category Axis Tick Label 包括 3 个子属性。

- Category Axis Tick Label Rotation.

- Category Axis Tick Label Font.
- Category Axis Tick Label Color.

前面讲过 Category Axis Vertical Tick Labels 属性并没有起作用, 如果想实现 Tick 的旋转效果, 则在 iReport 版本中可以使用 Category Axis Tick Label Rotation 属性实现, 报表的属性设置如图 6.73 所示。

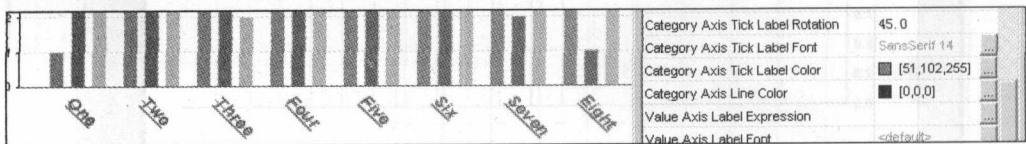


图 6.73 设置图表属性

运行效果如图 6.74 所示。

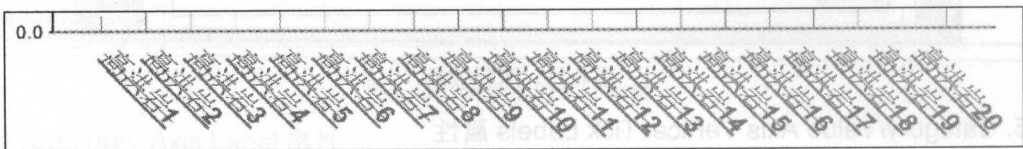


图 6.74 运行效果

7. Category/Value Axis Line Color 属性

属性 Category/Value Axis Line Color 的主要作用是设置 Category 和 Value 轴线的颜色, 设置 Category Axis Line Color 的颜色为红色, 设置 Value Axis Line Color 的颜色为蓝色, 运行效果如图 6.75 所示。

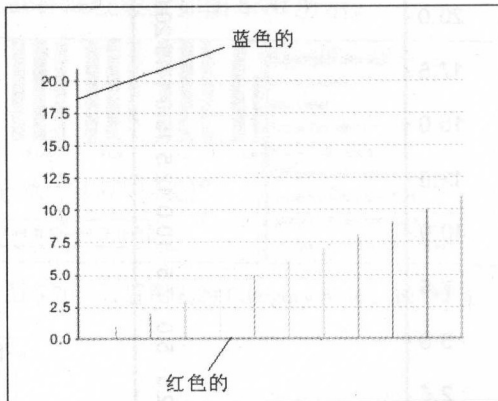


图 6.75 运行效果

8. Value Axis Label 属性

属性 Value Axis Label 包括 3 个子属性。

- Value Axis Label Expression: 用于设置值轴显示文字的表达式。
- Value Axis Label Font: 用于设置值轴显示文字的字体样式。

- Value Axis Label Color: 用于设置值轴显示文字的字体颜色。

报表属性的设置如图 6.76 所示。

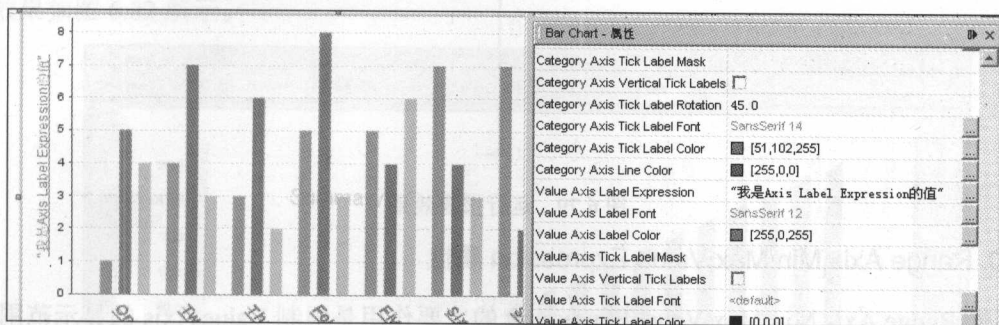


图 6.76 设置报表属性

运行报表，执行效果如图 6.77 所示。

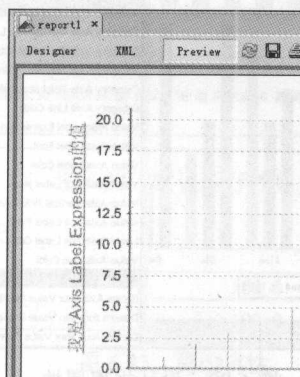


图 6.77 运行报表的效果

9. Value Axis Tick Label 属性

属性 Value Axis Tick Label 包括两个子属性。

- Value Axis Tick Label Font: 用于设置值轴 Tick 的字体样式。
- Value Axis Tick Label Color: 用于设置值轴 Tick 的字体颜色。

报表的属性设置如图 6.78 所示。

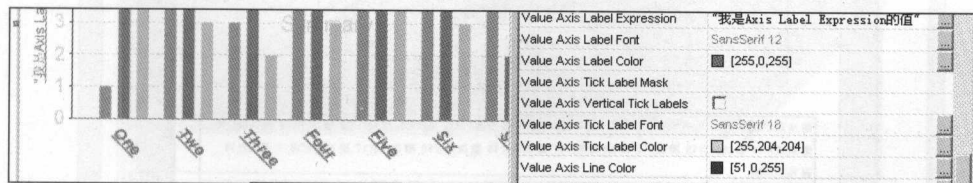


图 6.78 设置报表属性

运行报表，效果如图 6.79 所示。

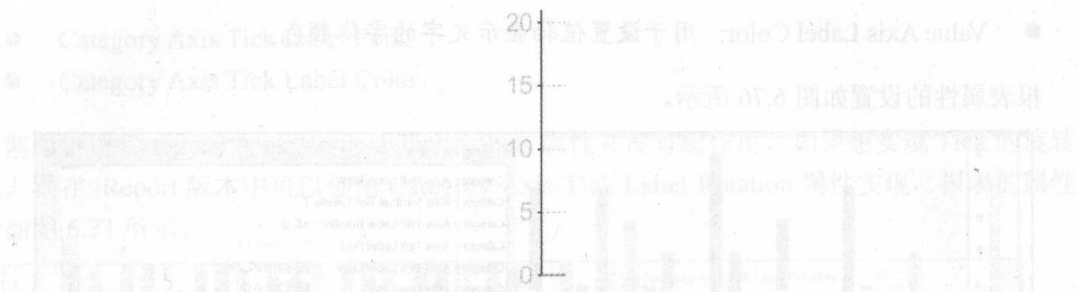


图 6.79 运行报表效果

10. Range Axis Min/Max Value Expression 属性

属性 Range Axis Min/Max Value Expression 的主要作用是限制 Value Axis 的显示范围，报表的属性设置如图 6.80 所示。

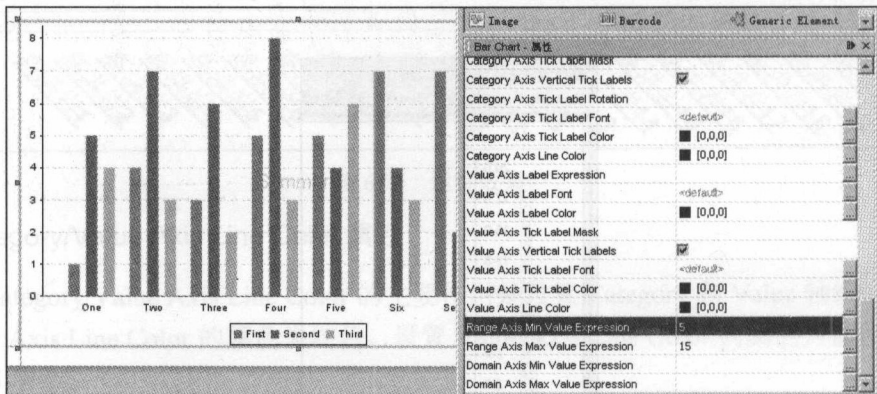


图 6.80 设计报表属性

程序的运行效果如图 6.81 所示。

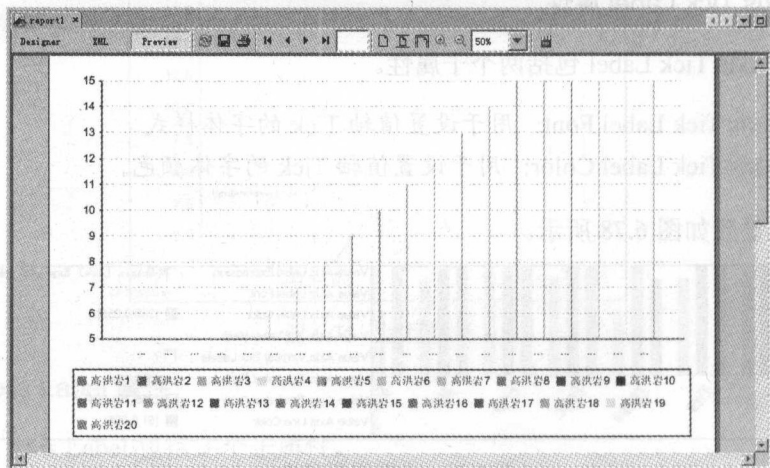


图 6.81 执行报表效果

从图 6.81 中可以看到 Value Axis 的范围被限制在 5~15 之间。

11. X/Y Offset 属性

属性 X Offset 和 Y Offset 的主要作用是定义带有立体效果的柱状图中柱的视觉样式,默认的效果如图 6.82 所示。

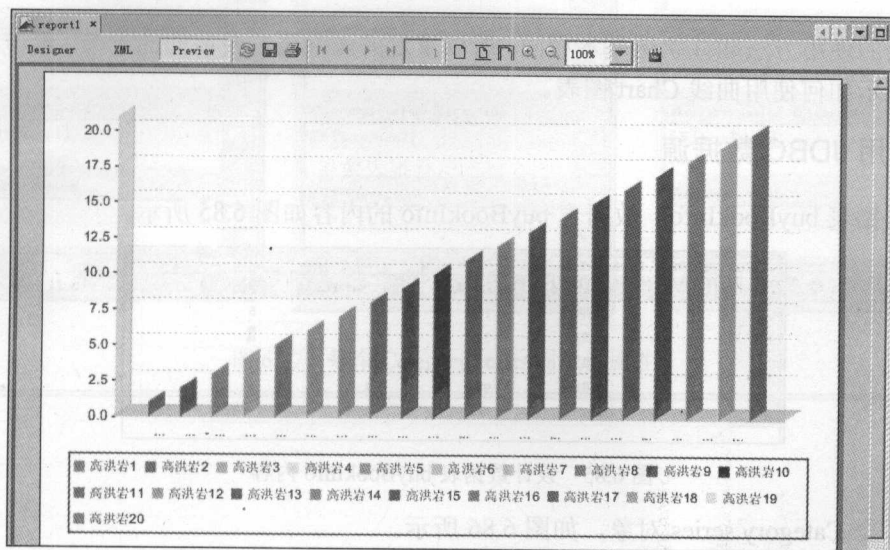


图 6.82 默认运行效果

更改属性设置如图 6.83 所示。

X Offset	100.0
Y Offset	100.0

图 6.83 更改属性

更改属性后的运行效果如图 6.84 所示。

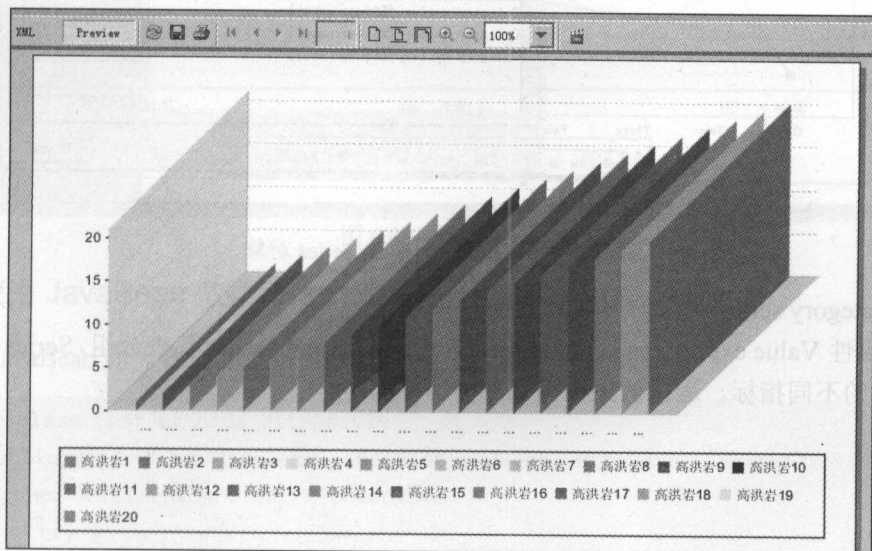


图 6.84 更改属性后的运行效果

至此已经将 5 种柱状图的常用属性都介绍完毕了。

6.3 Chart 图表的使用——曲线图

曲线图的使用方法和柱状图基本相同，所以重复的属性请参看前面的章节。本节将利用两个实验来演示如何使用曲线 Chart 图表。

6.3.1 使用 JDBC 数据源

设计数据表 buyBookInfo，数据表 buyBookInfo 的内容如图 6.85 所示。

TC05\SQL200... buyBookInfo					
	id	username	buyNum1	buyNum2	buyNum3
	1	高洪岩1	1	2	5
	2	高洪岩2	3	2	1
	3	高洪岩3	1	1	1
	4	高洪岩4	9	9	9
*	NULL	NULL	NULL	NULL	NULL

图 6.85 设计数据表 buyBookInfo 内容

添加 3 个 Category series 对象，如图 6.86 所示。

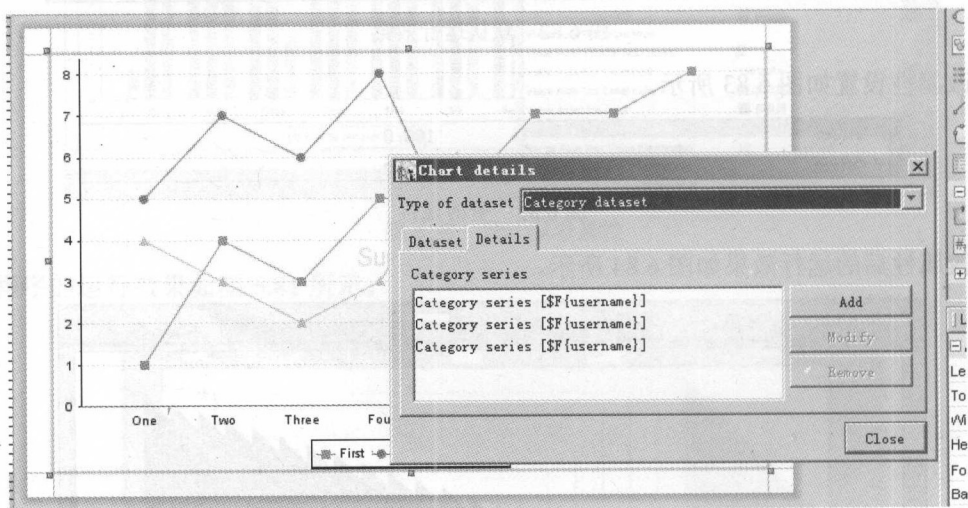


图 6.86 添加 3 个 Category series 对象

每个 Category series 对象的属性如图 6.87 所示。

其中，属性 Value expression 表示数据的值，Category expression 代表分组，Series expression 代表分组中的不同指标。运行效果如图 6.88 所示。

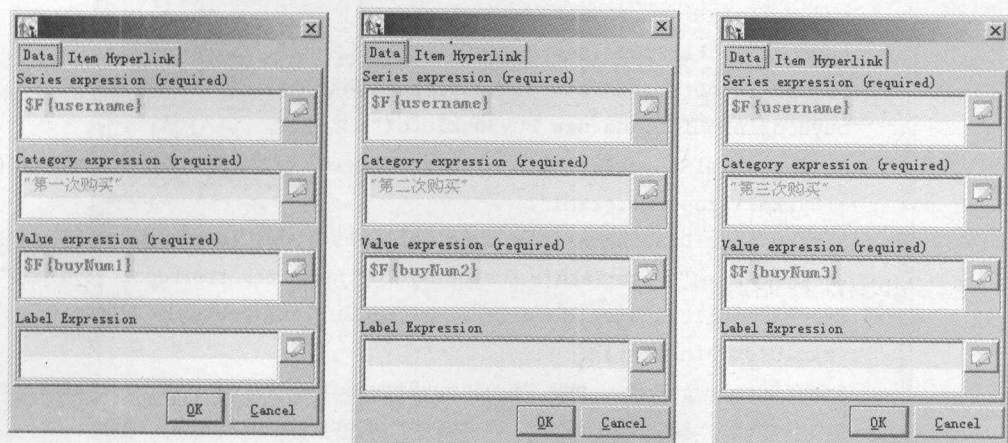


图 6.87 每个 Category series 对象的属性

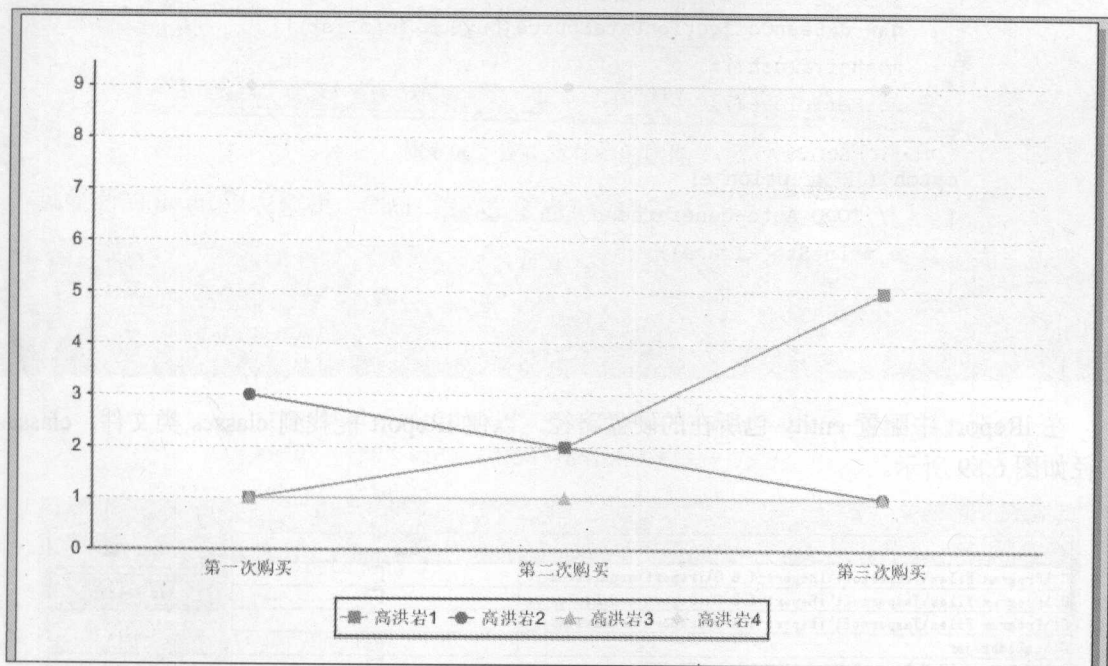


图 6.88 运行效果

6.3.2 使用 JavaBean 数据源

使用 JavaBean 作为数据源也很简单，创建 Servlet 的核心代码如下：

```
public class test extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List buyBookInfoList = new ArrayList();
```

```

        buyBookInfoList.add(new BuyBookInfo("高洪岩 1", 1, 2, 8));
        buyBookInfoList.add(new BuyBookInfo("高洪岩 2", 1, 2, 3));
        buyBookInfoList.add(new BuyBookInfo("高洪岩 3", 3, 2, 1));
        buyBookInfoList.add(new BuyBookInfo("高洪岩 4", 7, 7, 7));

        String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
        + "jrxml\\report1.jrxml";
        String jrxmlDestPathMain = this.getClass().getClassLoader()
        .getResource("").getPath().substring(1)+"jasperreports/report1.jasper";
        JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
        jrxmlDestPathMain);
        InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
        ServletOutputStream sosRef = response.getOutputStream();
        response.setContentType("application/pdf");
        JasperRunManager.runReportToPdfStream(isRef, sosRef, new HashMap(),
        new JRBeanCollectionDataSource(buyBookInfoList));
        sosRef.flush();
        sosRef.close();
    }

    catch (JRException e)
    { // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

在 iReport 中配置 entity 包所在的硬盘路径, 以便 iReport 能找到 classes 类文件, classes 路径如图 6.89 所示。

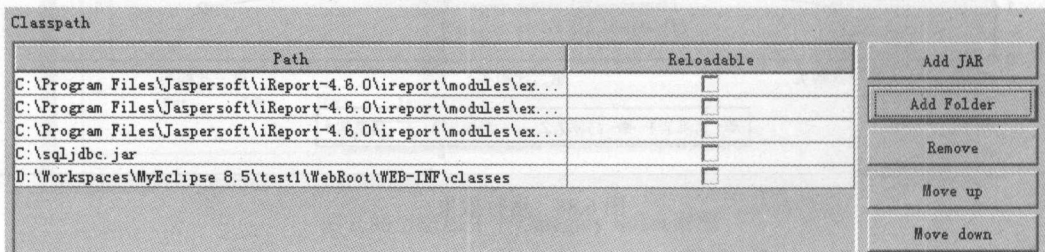


图 6.89 配置 classes 路径

逆向实体类的属性如图 6.90 所示。

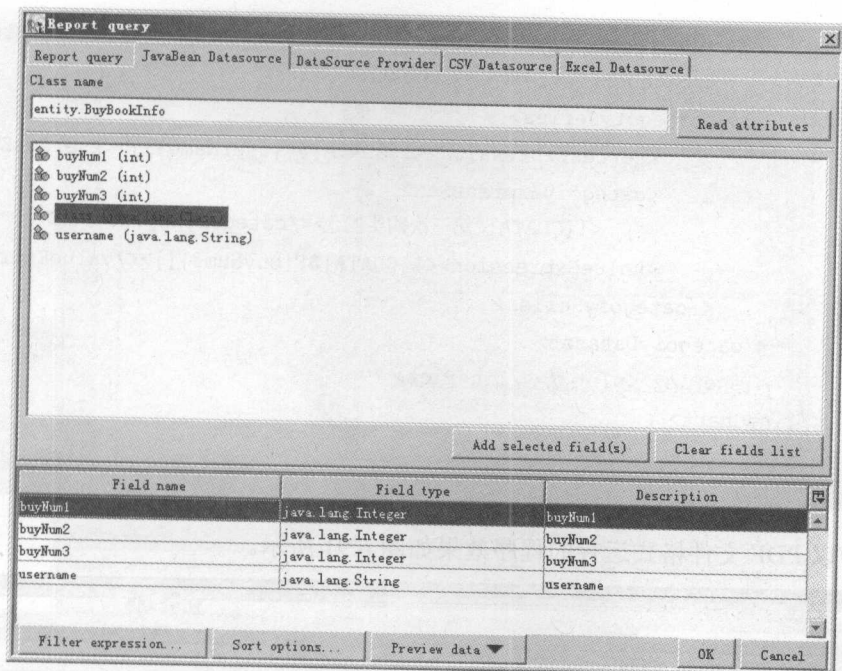


图 6.90 逆向实体类的属性

创建新的.jrxml 报表模板，Chart 核心配置代码如下：

```
<summary>
  <band height="323" splitType="Stretch">
    <lineChart>
      <chart>
        <reportElement uuid="895cd62e-c479-42e3-88dc-cbdd6cae1efc"
          x="0" y="0" width="555" height="323" />
        <chartTitle />
        <chartSubtitle />
        <chartLegend />
      </chart>
      <categoryDataset>
        <categorySeries>
          <seriesExpression><![CDATA[${username}]]></seriesExpression>
          <categoryExpression>
            <![CDATA["第一次购买"]]></categoryExpression>
          <valueExpression><![CDATA[${buyNum1}]]></valueExpression>
        </categorySeries>
        <categorySeries>
          <seriesExpression><![CDATA[${username}]]></seriesExpression>
          <categoryExpression>
            <![CDATA["第二次购买"]]></categoryExpression>
          <valueExpression><![CDATA[${buyNum2}]]></valueExpression>
        </categorySeries>
      </categoryDataset>
    </lineChart>
  </band>
</summary>
```

```

        <valueExpression><![CDATA[${buyNum2}]]></valueExpression>
    </categorySeries>
    <categorySeries>
        <seriesExpression><![CDATA[${username}]]></seriesExpression>
        <categoryExpression>
            <![CDATA["第三次购买"]]></categoryExpression>
        <valueExpression><![CDATA[${buyNum3}]]></valueExpression>
    </categorySeries>
</categoryDataset>
<linePlot><plot /></linePlot>
</lineChart>
</band>
</summary>

```

在 IE 中以 PDF 文件格式运行的程序效果如图 6.91 所示。

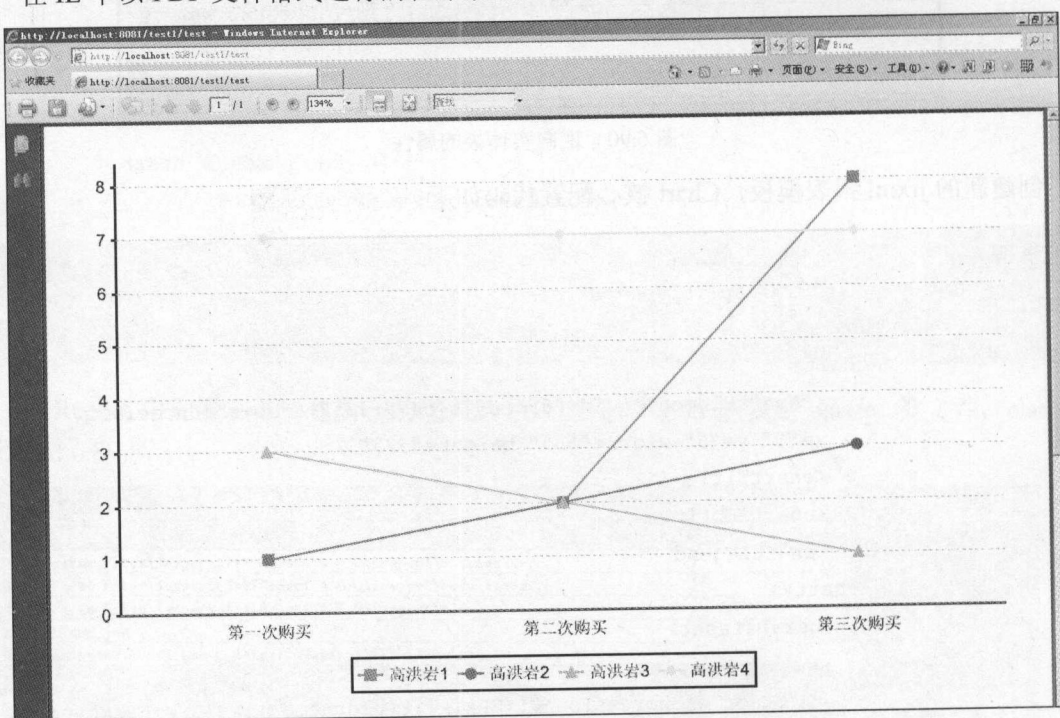


图 6.91 在 IE 中以 PDF 文件格式运行的效果

6.4 在图表 Chart 中添加超链接

为饼状图配置数据源，如图 6.92 所示。

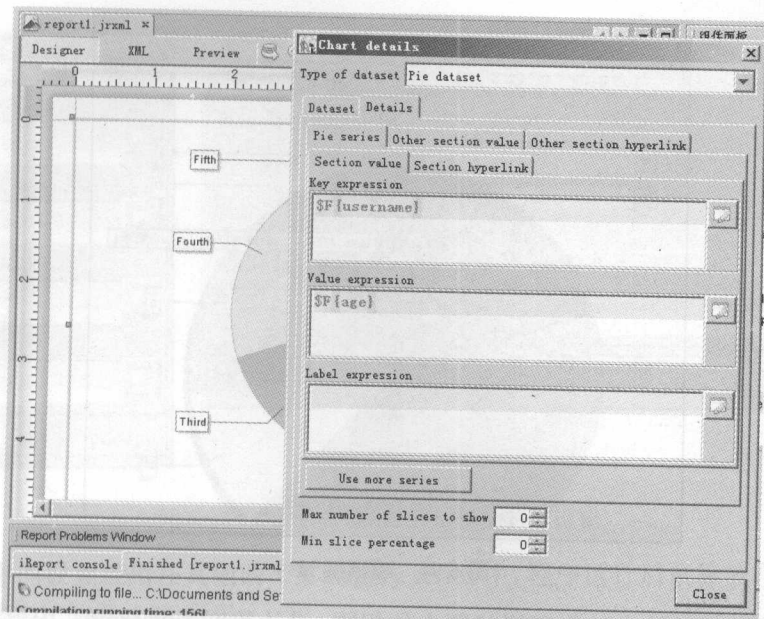


图 6.92 配置饼状图数据源

设置饼状图的超链接选项，如图 6.93 所示。

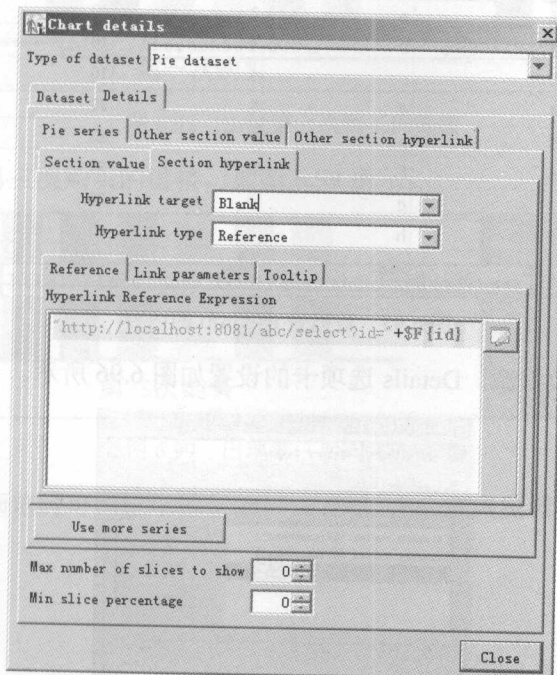


图 6.93 设置超链接

报表运行后不仅显示出了饼状图，还在每一个分片中显示出了超链接（id=1），如图 6.94 所示。

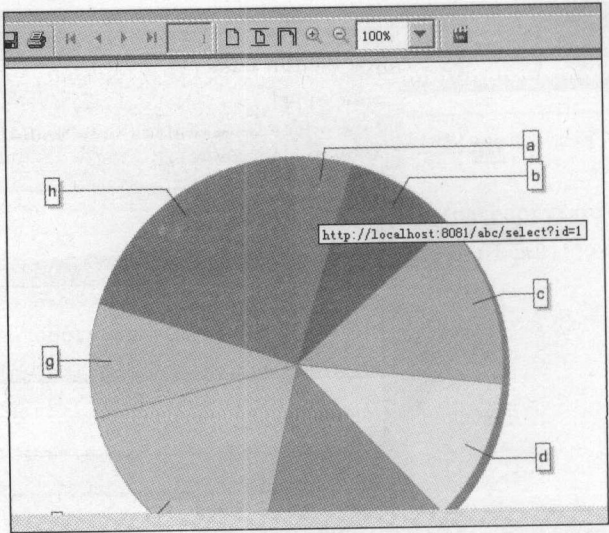


图 6.94 超链接 id=1

设置柱状图的超链接方法也和饼状图大体相同，设计数据表 userinfo，内容如图 6.95 所示。

CS212\SQL20...bo.userinfo				
	id	username	age	usertype
	1	a	2	10
	2	b	4	10
	3	c	6	10
	4	d	5	20
	5	e	7	20
	6	f	8	30
	7	g	4	30
	8	h	9	40
*	NULL	NULL	NULL	NULL

图 6.95 数据表 userinfo 的内容

设置柱状图的图表数据源，Details 选项卡的设置如图 6.96 所示。

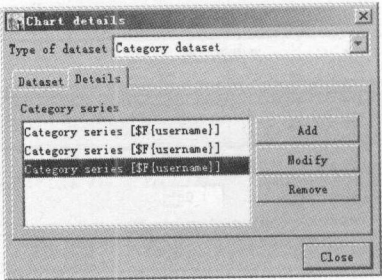


图 6.96 Details 选项卡的设置

Category series 和链接的具体设置如图 6.97 所示。

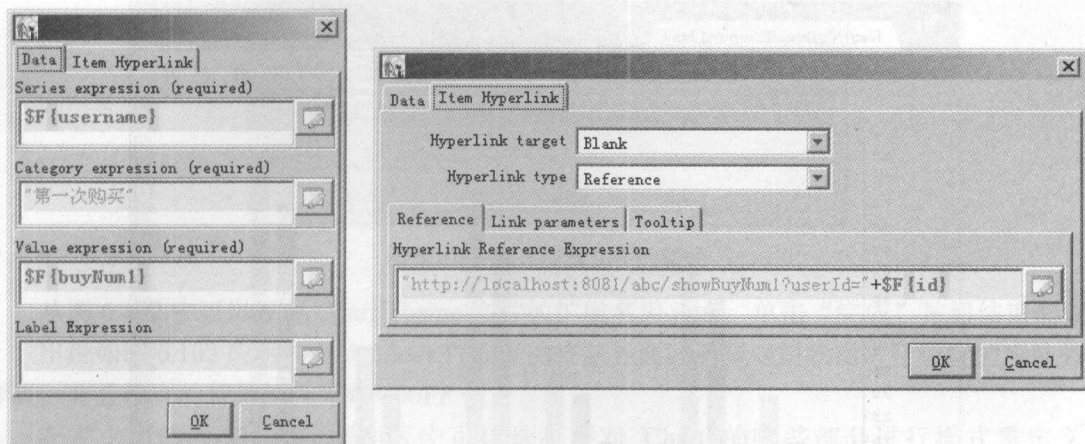


图 6.97 Category series 和链接的具体设置

将图 6.97 中的设置应用在其他两个 Category series 中, 程序运行后鼠标放在图表上将出现第一次购买的链接, 如图 6.98 所示。

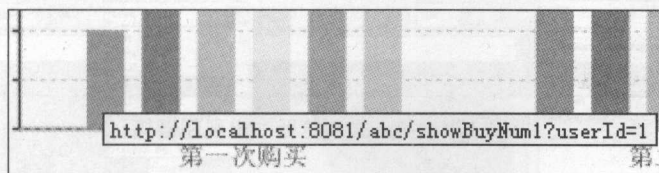


图 6.98 出现第一次购买的链接

继续测试, 出现第三次购买的链接, 如图 6.99 所示。

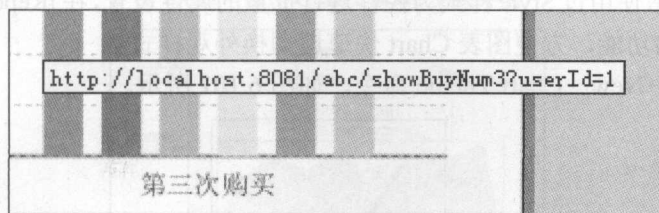


图 6.99 出现第三次购买的链接

利用 HTML 格式预览时可以出现链接的样式, 例如在火狐浏览器中出现超链接, 如图 6.100 所示。

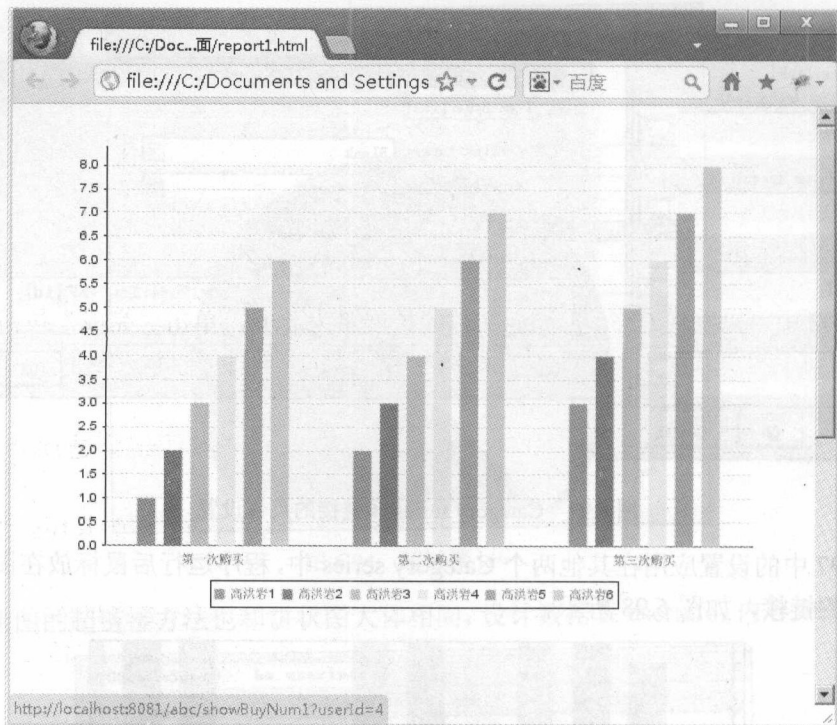


图 6.100 在火狐浏览器中出现超链接

6.5 在图表 Chart 中使用皮肤 Themes

在前面的章节中使用过 Style 样式为控件进行批量的属性设置,在 iReport 中还可以对 Chart 应用皮肤 Themes 的功能,方便图表 Chart 快速地切换外观样式。

选择“文件”→New→Chart Theme 命令,如图 6.101 所示。

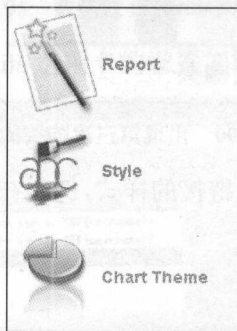


图 6.101 选择 Chart Theme

单击 Finish 按钮继续下一步操作。

出现保存皮肤文件的路径,如图 6.102 所示。

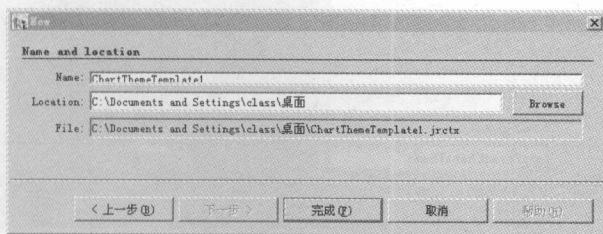



图 6.102 保存皮肤文件的路径

从图 6.102 中可以发现 Chart Theme 文件的扩展名为.jrctx，单击“完成”按钮结束配置。

出现如图 6.103 所示的设计 Chart Theme 的皮肤界面。选择对饼状图进行设置的皮肤外观样式，并且设置的背景颜色为#CC00FF 值。

在左上角的 Template Inspector 面板中可以选择要对 Chart 的哪些部分进行样式重定义，Template Inspector 结构与图表内容示意图如图 6.104 所示。

皮肤模板设置完毕后将此皮肤导出为.jar 文件，即单击  按钮，如图 6.105 所示。选择导出的路径，如图 6.106 所示。

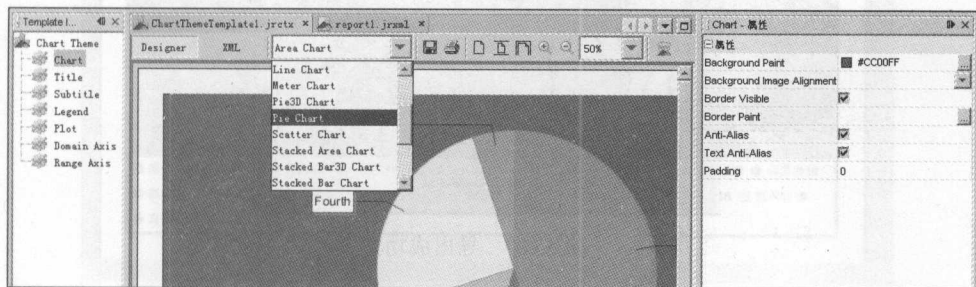


图 6.103 设计 Chart Theme 皮肤界面

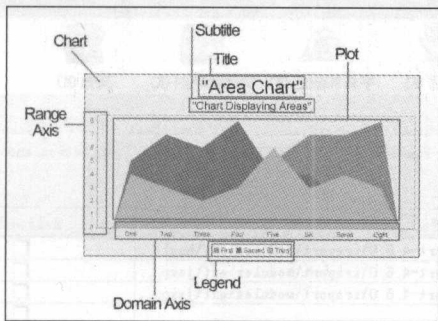


图 6.104 Template Inspector 结构与图表内容示意图

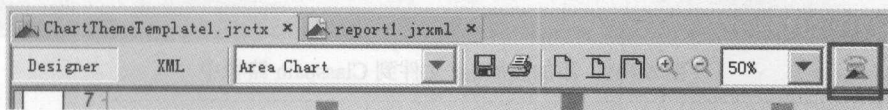


图 6.105 导出.jar 文件

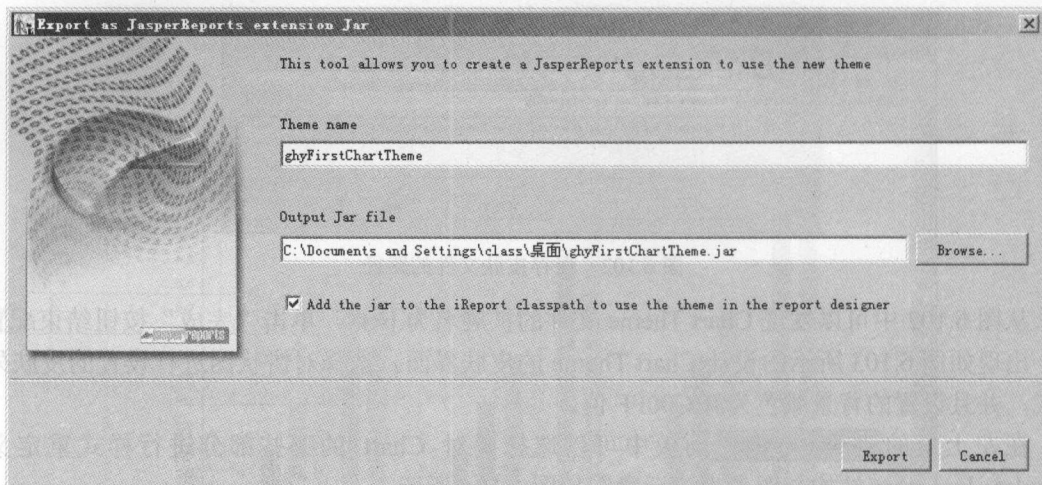


图 6.106 选择导出的路径

弹出导出成功的提示, 如图 6.107 所示。

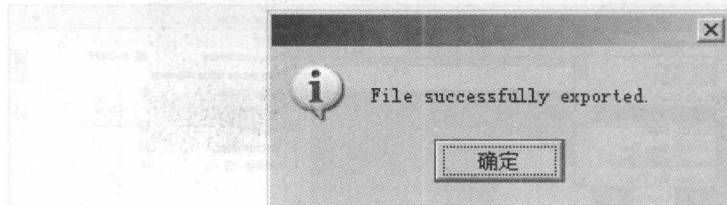


图 6.107 导出成功

添加皮肤.jar 文件到 Classpath 路径中, 如图 6.108 所示。

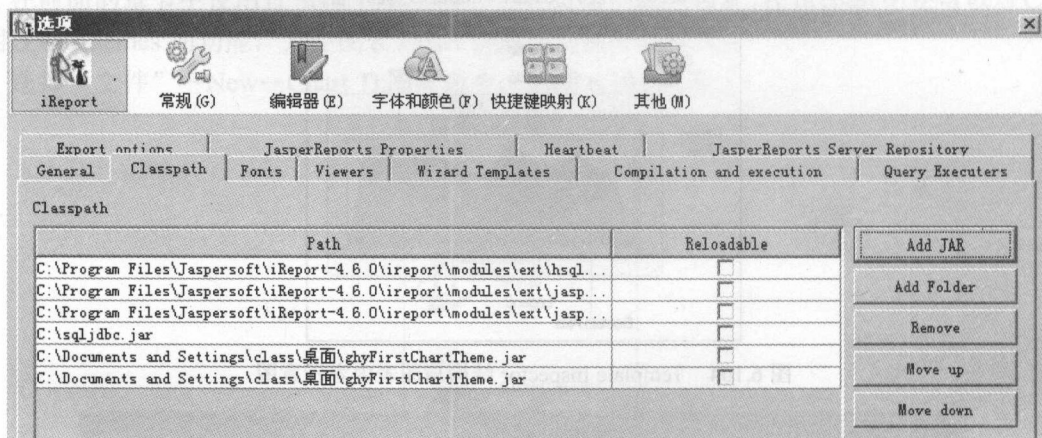


图 6.108 添加皮肤.jar 文件到 Classpath 路径中

新建一个报表.jrxml 模板, 将 Theme 设置为 ghyTheme, 如图 6.109 所示。

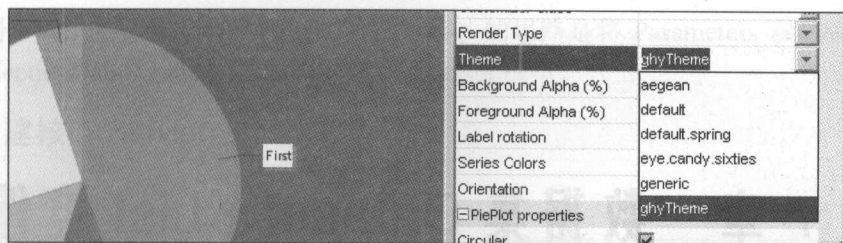


图 6.109 选择 ghyTheme

程序运行后的效果如图 6.110 所示，即成功添加皮肤背景为紫色。

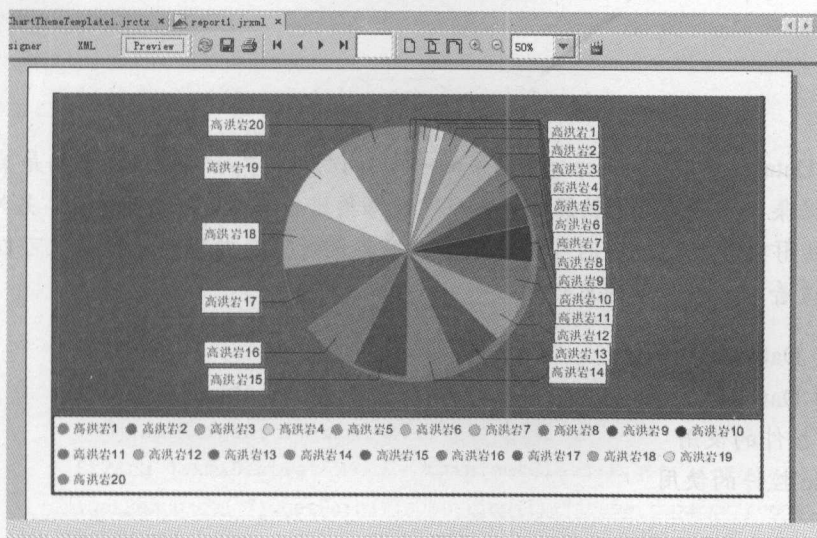


图 6.110 成功添加皮肤背景为紫色

第 7 章 数据集 Dataset、List 控件 及 Table 控件



引言

数据集 Dataset 是 JasperReports 框架的一个知识点，学习它的主要目的是某些控件不能使用主数据集，而要使用子数据集，从而对数据集进行有效、分类的管理，另外控件 List 和 Table 的使用非常重要，它们可以结合子数据集进行报表内容的填充，也可以替代子报表，读者应该着重掌握如下内容：

- ★ 创建 Dataset 数据集
- ★ 编辑 Dataset 数据集的数据源，可以是 SQL 或 JavaBean
- ★ List 控件的使用
- ★ Table 控件的使用

7.1 数据集 Dataset

在前面的章节中，如果想在报表中循环打印数据源中的数据，则必须在代码：“JasperRunManager.runReportToPdfStream(isRef,sosRef, new HashMap(),new JRBeanCollectionDataSource(buyBookInfoList1));”的最后一个参数将 List 对象转成 JRBeanCollectionDataSource 对象，再传递给.jasper 文件才可以打印出来，那如果想要一起打印两个数据源怎么办？如在 Summary 栏中有两个 Chart 控件需要一起输出不同数据源中的数据，那么上面的代码就不可能办到，因为只提供了一个数据源，回忆一下在前面学习过的知识，使用 Subreport 就可以解决这个问题，解决的办法是将 List 以 Parameters 参数的形式传递给报表，然后在报表中使用 Subreport 控件将这个 List 参数转成 JRBeanCollectionDataSource 对象作为数据源，示例代码如下：

```
new net.sf.jasperreports.engine.data.JRBeanCollectionDataSource  
($P{param_userinfo}.getListAddress())
```

但使用 Subreport 毕竟是要增加.jrxml 文件的数量，其实使用 Dataset 就可以解决这个问题。本章将和大家一起讨论如何将 Parameters 参数中的 List 变成多个数据源，然后在两个 Chart

控件中使用不同的数据源来打印不同的数据，解决思路是将 Parameters 参数中的 List 转成 JRBeanCollectionDataSource 对象，再传递给 Dataset 即可。

7.1.1 创建核心 Servlet

核心代码如下：

```
public class test extends HttpServlet
{
    JRBeanCollectionDataSource test;

    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List buyBookInfoList1 = new ArrayList();
            for (int i = 0; i < 10; i++)
            {
                buyBookInfoList1.add(new BuyBookInfo("username1-"+(i+1), (i +1)));
            }
            List buyBookInfoList2 = new ArrayList();
            for (int i = 10; i < 21; i++)
            {
                buyBookInfoList2.add(new BuyBookInfo("username2-"+(i+1), (i+1)));
            }
            String jrxmlSourcePathMain = this.getServletContext().getRealPath("/")
                + "jrxml\\report1.jrxml";
            String jrxmlDestPathMain = this.getClass().getClassLoader()
                .getResource("").getPath().substring(1)+"jasperreports/report1.jasper";
            JasperCompileManager.compileReportToFile(jrxmlSourcePathMain,
                jrxmlDestPathMain);
            InputStream isRef = new FileInputStream(new File(jrxmlDestPathMain));
            ServletOutputStream sosRef = response.getOutputStream();
            response.setContentType("application/pdf");
            HashMap paramMap = new HashMap();
            paramMap.put("buyBookInfoList1", buyBookInfoList1);
            paramMap.put("buyBookInfoList2", buyBookInfoList2);
            JasperRunManager.runReportToPdfStream(isRef, sosRef, paramMap,
                new JREmptyDataSource());
            sosRef.flush();
            sosRef.close();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

对象 paramMap 中有两个 key，分别是 buyBookInfoList1 和 buyBookInfoList2，所以还要在报表中创建两个 Parameters 参数对象来接收这两个 List。

7.1.2 创建报表模板

创建报表模板，在 Summary 栏中添加两个 Chart 饼状图控件，如图 7.1 所示。

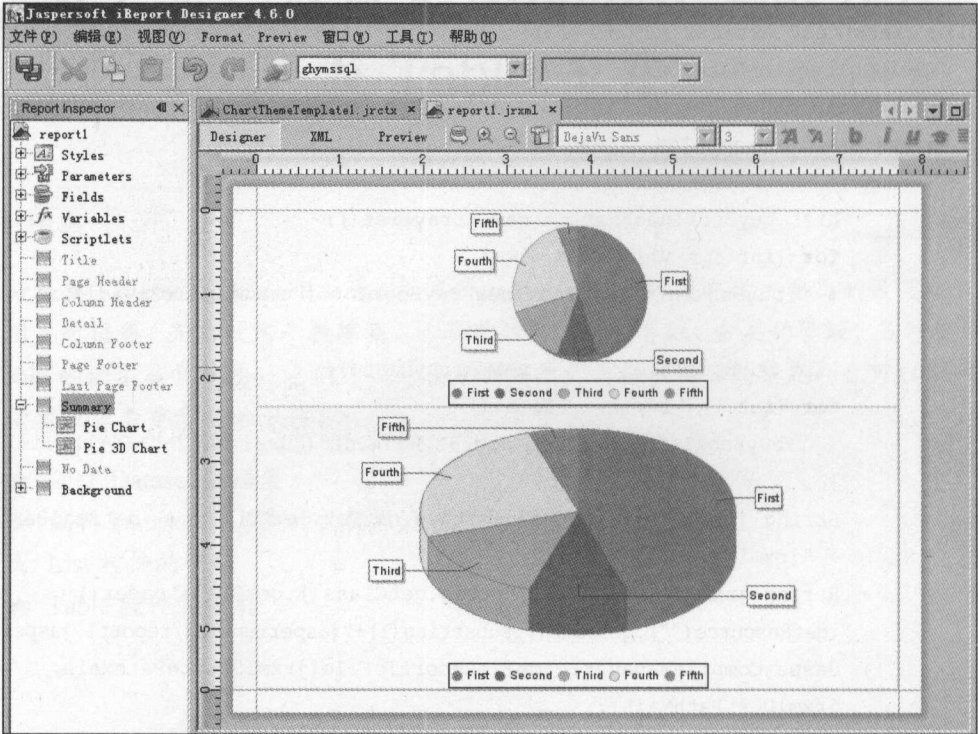


图 7.1 添加两个饼状图控件

7.1.3 创建 Dataset 数据集

添加一个 Dataset 数据集，如图 7.2 所示。

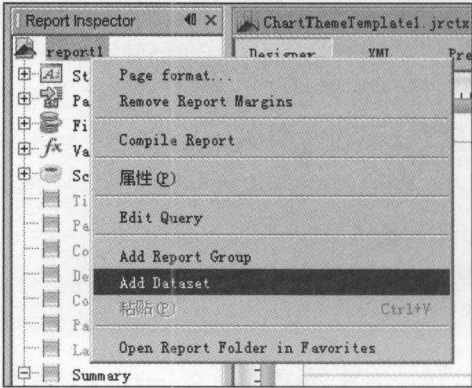


图 7.2 添加一个 Dataset 数据集

创建一个空的数据集，如图 7.3 所示。

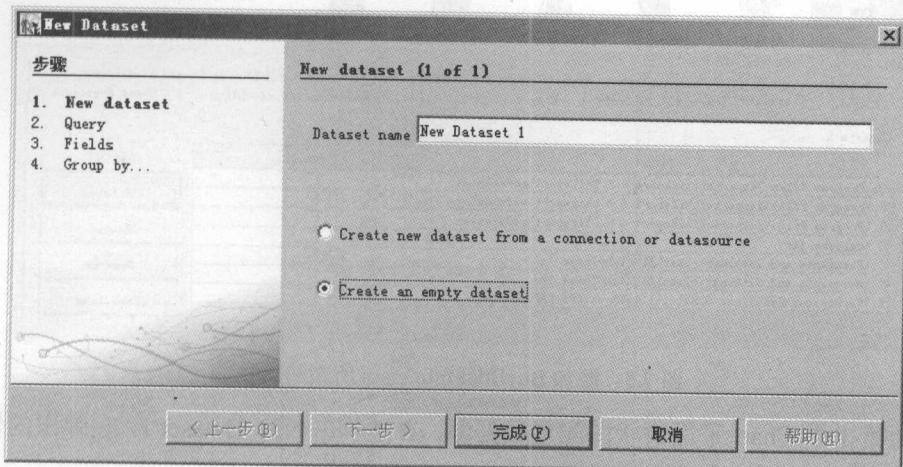


图 7.3 创建一个空的数据集

创建空的数据集的主要目的是自己关联 List 中存储 Bean 的数据源，单击“完成”按钮。利用同样的方法再创建一个 Dataset 数据集，创建完成后的报表模板树结构如图 7.4 所示，即已成功添加两个 Dataset 数据集。

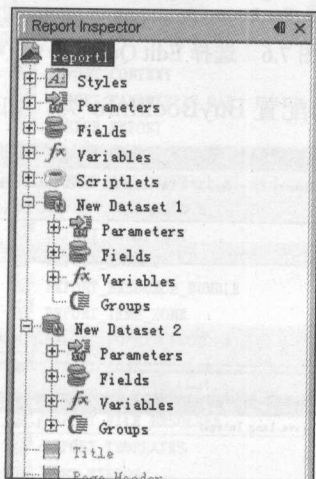


图 7.4 成功添加两个 Dataset 数据集

7.1.4 配置 Dataset 数据集

此时 Dataset 数据集中并没有数据，所以要关联 List 中存储的 JavaBean。

在 Classpath 中添加 JavaBean 的路径，本例中是添加 BuyBookInfo.class 所在的路径，如图 7.5 所示。

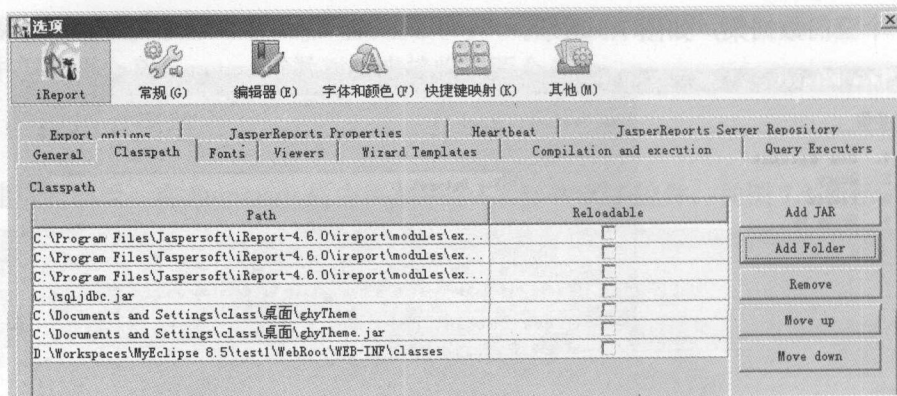


图 7.5 添加 BuyBookInfo.class 所在的路径

然后为第 1 个 Chart 平面饼状图配置数据源, 右键单击 New Dataset 1, 在弹出的快捷菜单中选择 Edit Query 命令, 如图 7.6 所示。

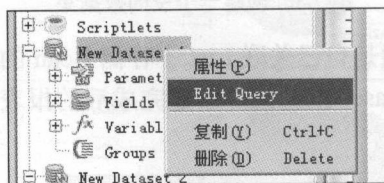


图 7.6 选择 Edit Query 命令

配置 JavaBean 选项, 本例中为配置 BuyBookInfo 类, 如图 7.7 所示。

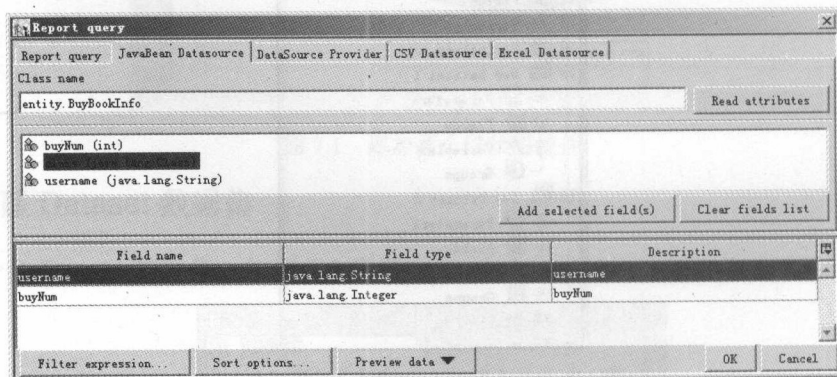


图 7.7 配置 BuyBookInfo 类

配置完成后在 Fields 节点中出现两个字段, 如图 7.8 所示。

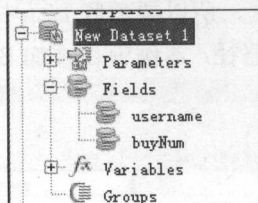


图 7.8 Fields 节点中出现两个字段

利用同样的方式配置第 2 个数据集，完成后的效果如图 7.9 所示。

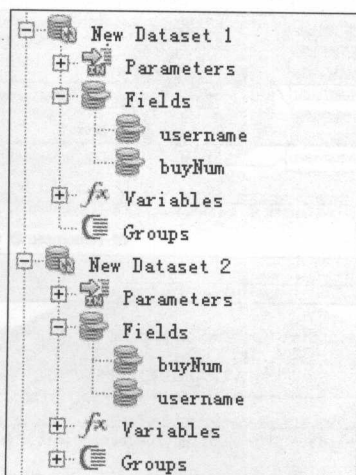


图 7.9 两个 Dataset 都有两个字段

在 report1.jrxml 中添加两个 Parameters 参数，如图 7.10 所示。

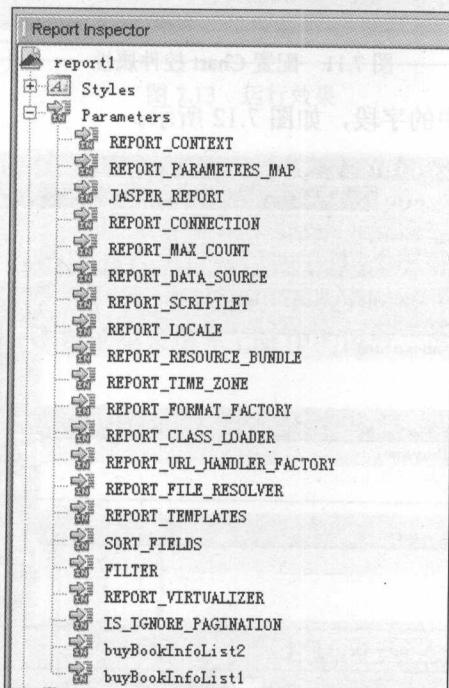


图 7.10 添加两个 Parameters 参数

7.1.5 关联 Dataset 数据集

配置第 1 个平面饼状图控件的属性，如图 7.11 所示。

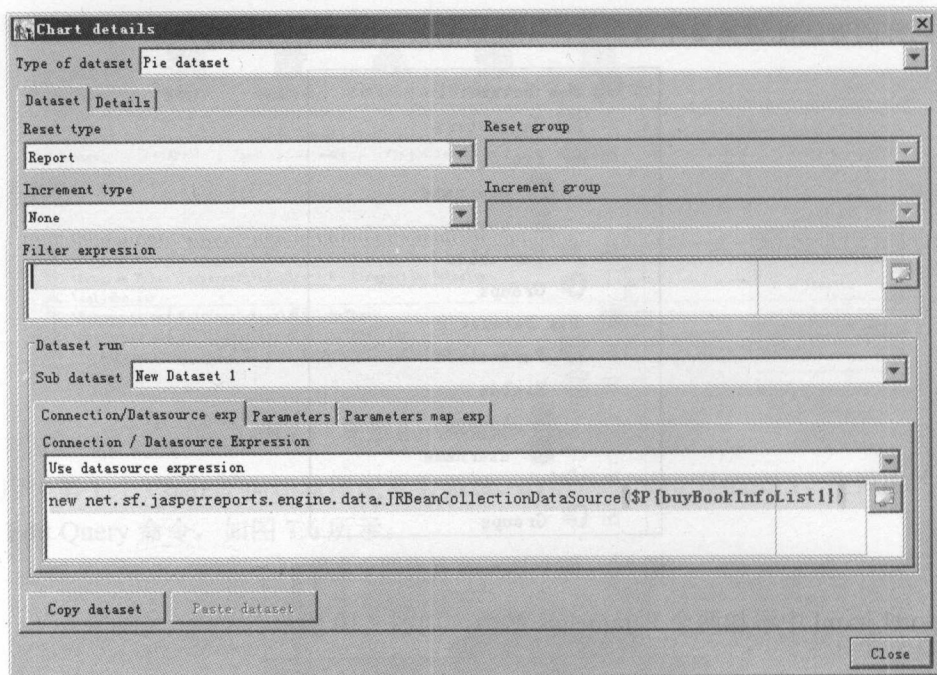


图 7.11 配置 Chart 控件属性

并且配置 Details 选项卡中的字段，如图 7.12 所示。

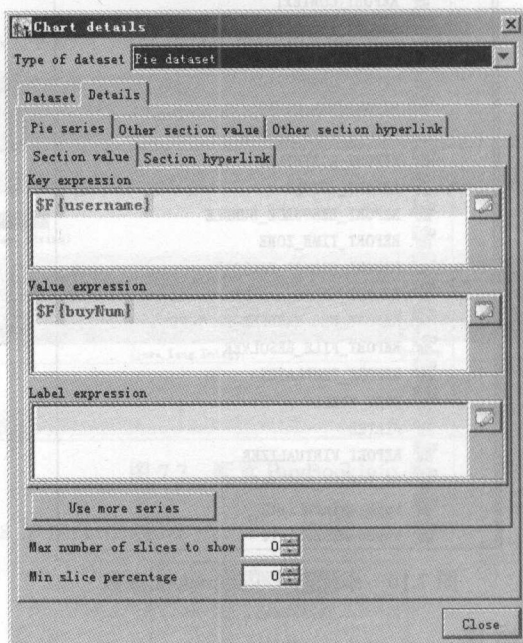


图 7.12 配置字段

利用同样的方法配置第 2 个图表控件的属性。

运行效果如图 7.13 所示。

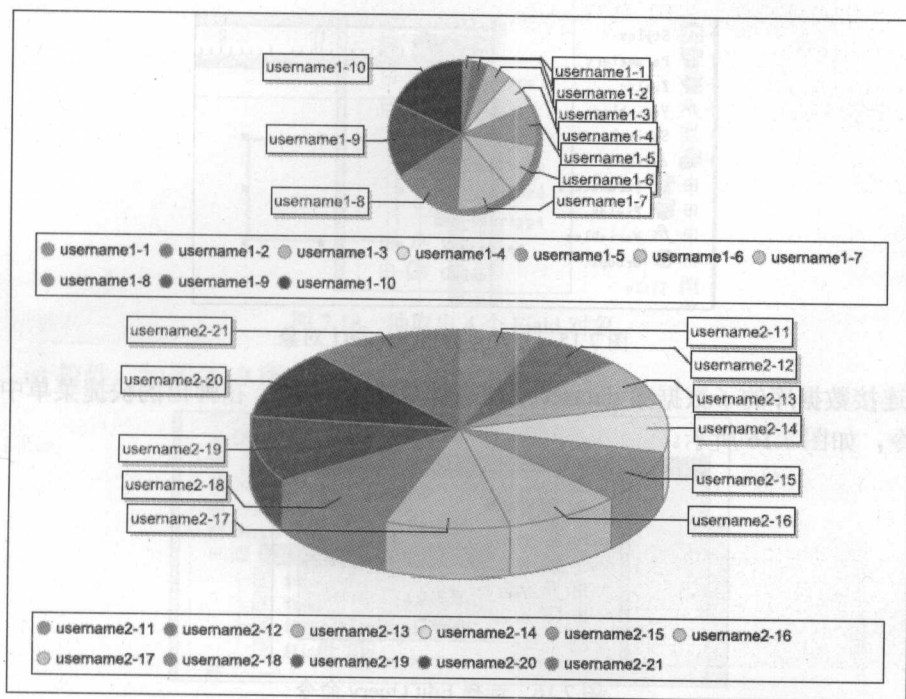


图 7.13 运行效果

7.2 List 控件

List 控件是一个轻量级的 Subreport 子报表控件, 可以不用单独的 .jrxml 来显示列表中的数据, 数据的内容可以有多种类, 如图片、文字等, 该控件的使用也非常简单, 需要注意的是, List 控件必须使用 SubDataSet 子数据集来填充 List 中的内容, 而不能通过 MainDataSet 主数据集来进行填充。

下面将介绍 List 控件使用 Connection 数据源进行打印测试的方法。在 Detail 1 栏中创建一个 List 控件, 如图 7.14 所示。

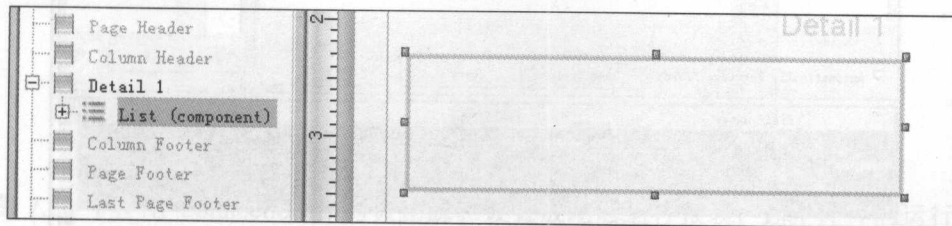


图 7.14 在 Detail 1 中添加 List 控件

当 List 控件被创建后将自动在左边的报表结构面板 Report Inspector 中添加一个 dataset1 子数据集对象, 但这个子数据集对象是空的, 所以可以通过 Connection 或 JavaBean 的方式来创建 Fields 对象, 效果如图 7.15 所示。

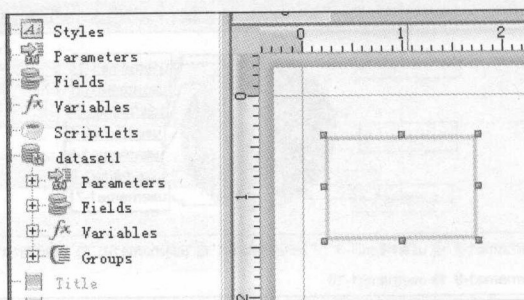


图 7.15 自动创建的 dataset1 对象

创建连接数据库的子数据集 dataset1，即右键单击 dataset1，在弹出的快捷菜单中选择 Edit Query 命令，如图 7.16 所示。

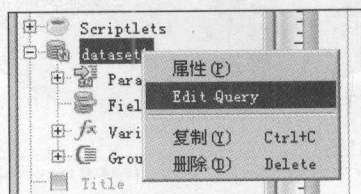


图 7.16 选择 Edit Query 命令

在弹出的对话框中输入 SQL 语句，如图 7.17 所示。

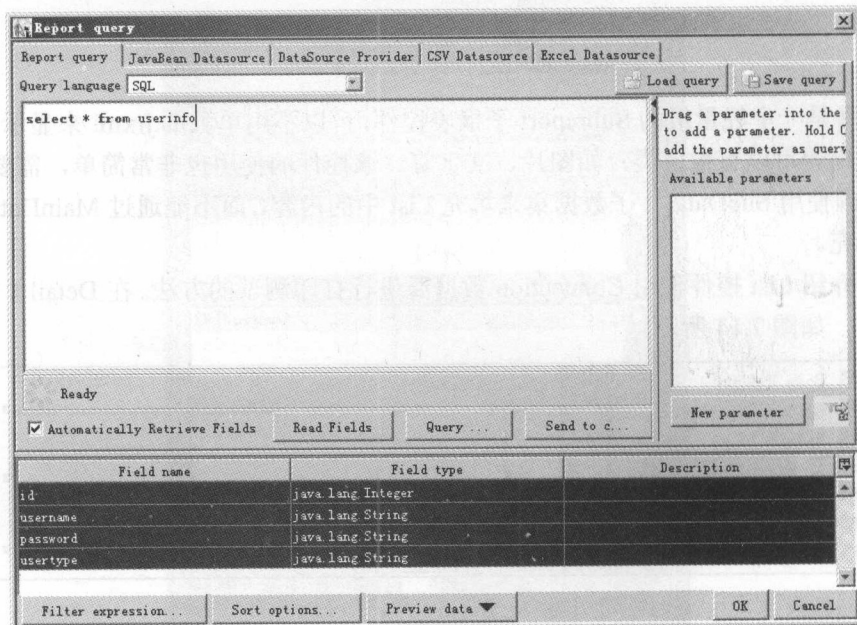


图 7.17 输入 SQL 语句

抽取 4 个 Field 对象，如图 7.18 所示。

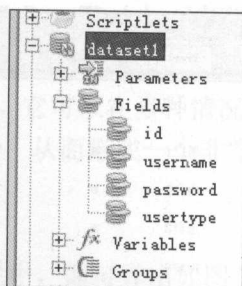


图 7.18 抽出 4 个 Field 对象

编辑 List 控件，如图 7.19 所示。

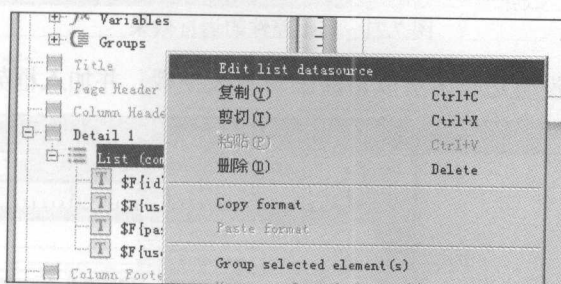


图 7.19 编辑 List 控件

设置关联 dataset1 数据集，如图 7.20 所示。

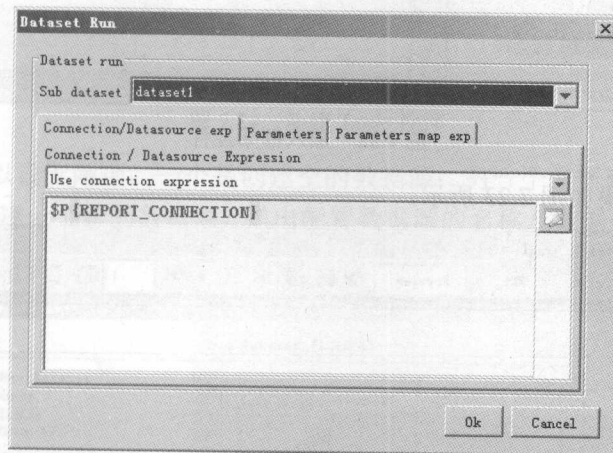
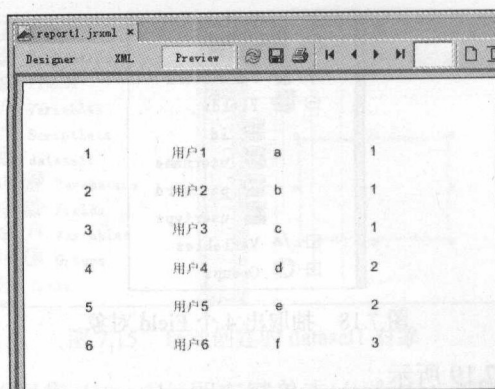


图 7.20 设置关联 dataset1 数据集

在图 7.20 中使用 Connection 作为数据源。设置成功后运行报表，List 控件的运行效果如图 7.21 所示。



1	用户1	a	1
2	用户2	b	1
3	用户3	c	1
4	用户4	d	2
5	用户5	e	2
6	用户6	f	3

图 7.21 List 控件的运行效果

此时的 List 控件并没有边框，所以重新编辑报表模板，并加入框架控件，如图 7.22 所示。

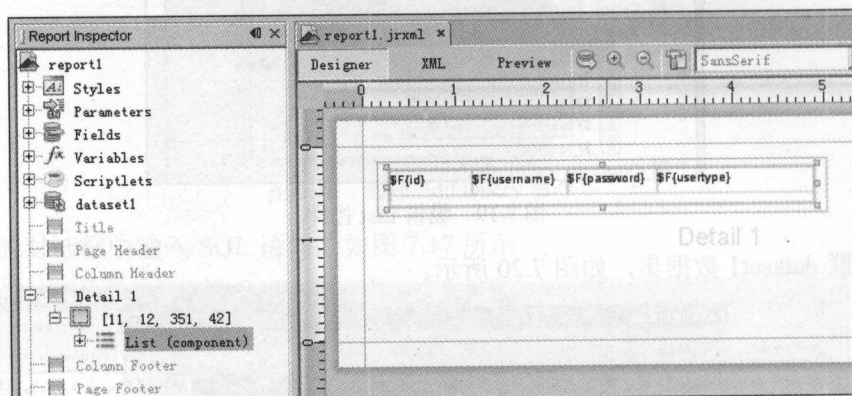
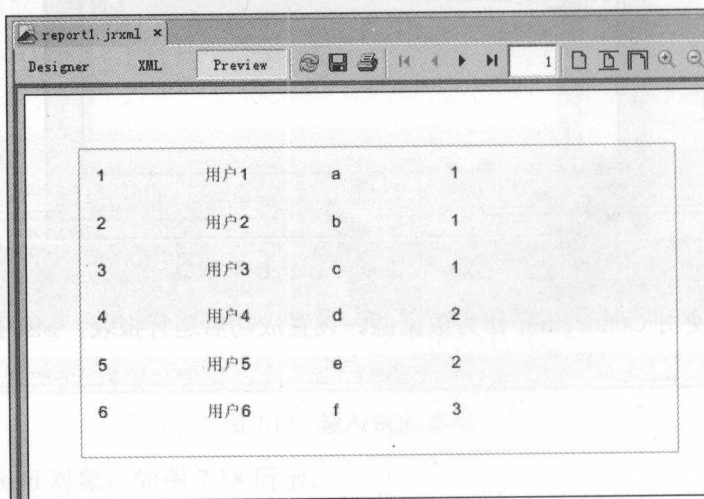


图 7.22 加入框架控件

有边框的运行效果如图 7.23 所示。



1	用户1	a	1
2	用户2	b	1
3	用户3	c	1
4	用户4	d	2
5	用户5	e	2
6	用户6	f	3

图 7.23 有边框的运行效果

7.3 Table 控件

控件 Table 是强有力的报表组件，它可以在多种情况下将子报表 Subreport 替代，Table 组件中可以存放其他多种报表组件元素，从而形成一个非常复杂的报表布局。

7.3.1 使用 Table 控件

将 Table 控件拖曳到 Summary 栏中，即可弹出如图 7.24 所示的新建 Table 控件对话框。

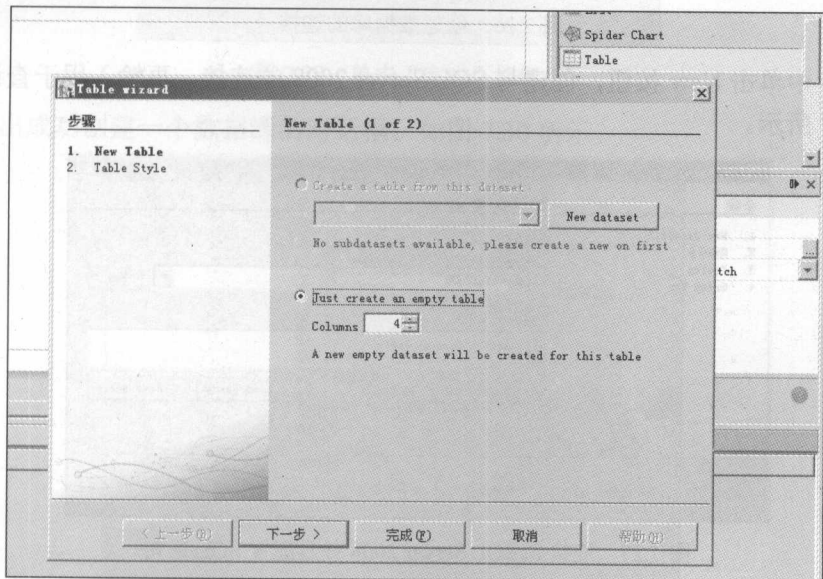


图 7.24 新建 Table 控件

在图 7.24 中可以设置两项，一个是创建空的数据源；另一个是新建一个数据源，在这里单击 **New dataset** 按钮创建一个新的数据源，弹出配置数据源的名称及数据来源方式的对话框，如图 7.25 所示。

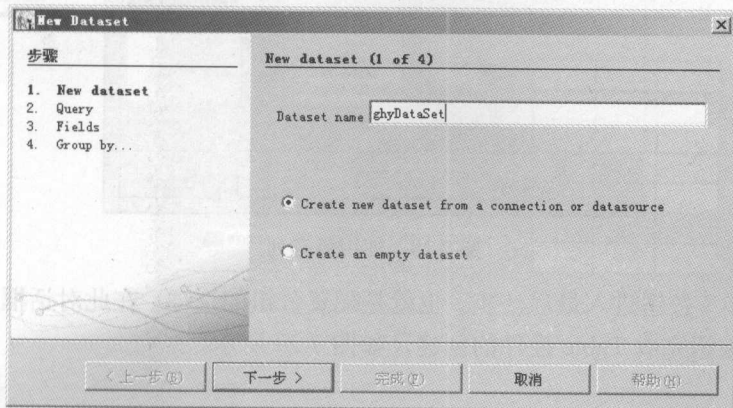


图 7.25 配置数据源的名称及数据来源方式

设置完成后，单击“下一步”按钮，出现数据库连接对话框，如图 7.26 所示。

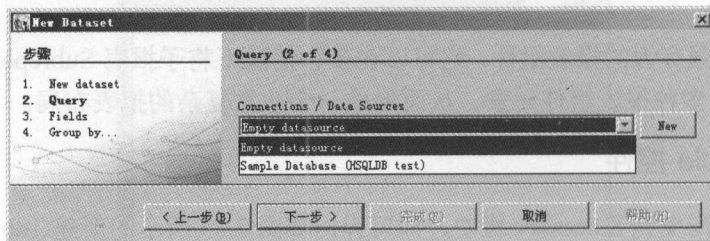


图 7.26 创建数据库的连接

在图 7.26 中单击 New 按钮，创建与 SQL Server 2008 的连接，再输入用于查询的 SQL 语句，如图 7.27 所示。

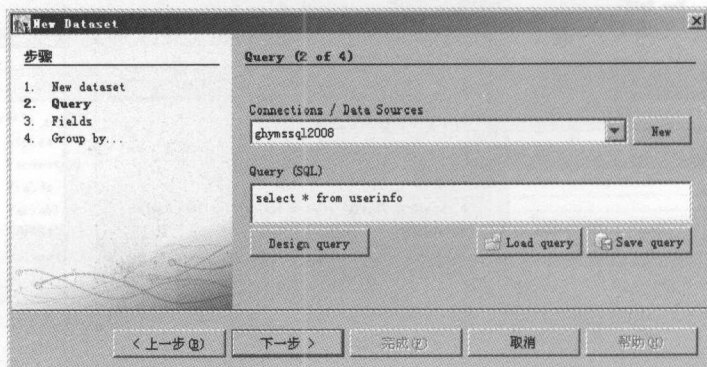


图 7.27 连接到 SQL Server 2008 并且配置 SQL 语句

单击“下一步”按钮继续配置，出现如图 7.28 所示的对话框。

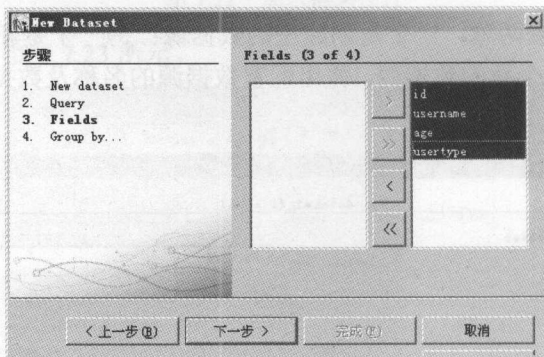


图 7.28 使用所有列

单击“下一步”按钮进入最后一步，也就是配置分组的选项，在此对话框中不做修改，直接单击“完成”按钮完成 Table 控件的创建，如图 7.29 所示。

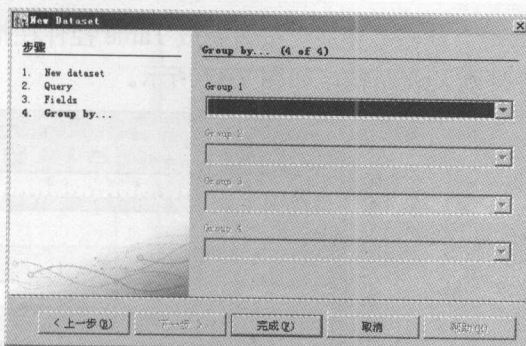


图 7.29 单击“完成”按钮

此时将弹出成功创建一个数据源的对话框，如图 7.30 所示。

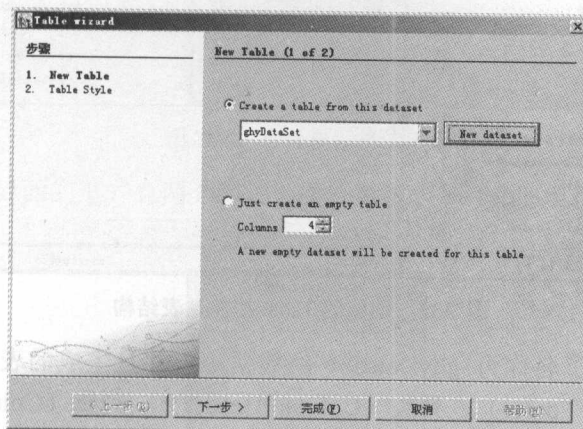


图 7.30 成功创建一个数据源

继续单击“下一步”按钮继续配置，弹出如图 7.31 所示的配置 Table 控件的外观对话框。

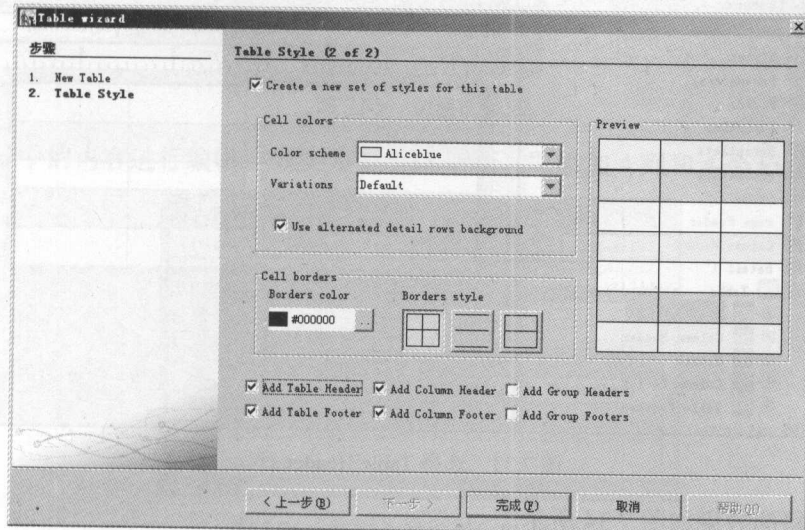


图 7.31 配置 Table 控件的外观

使用默认的外观配置即可，单击“完成”按钮完成 Table 控件在报表模板中的创建。新建在报表模板中的 Table 控件外观及表结构，如图 7.32 所示。

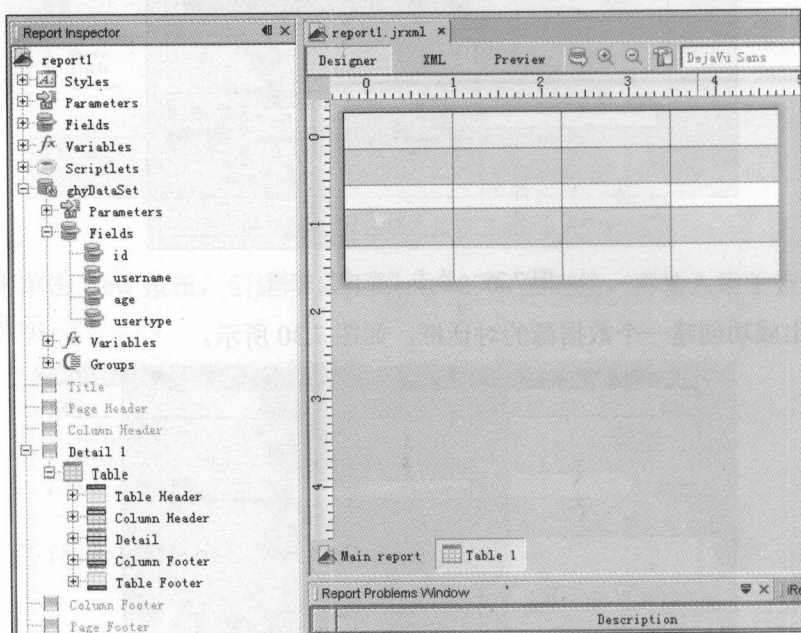


图 7.32 创建的 Table 控件及表结构

在图 7.32 中生成了名称为 ghyDataSet 的数据集，并且生成的表结构包括表头 (Table Header)、表脚 (Table Footer)、列头 (Column Header)、列脚 (Column Footer) 以及详细 (Detail) 5 个 Band 栏。

表格分为行和列，选择行的操作很简单，如图 7.33 所示为选择 Table Header 行。

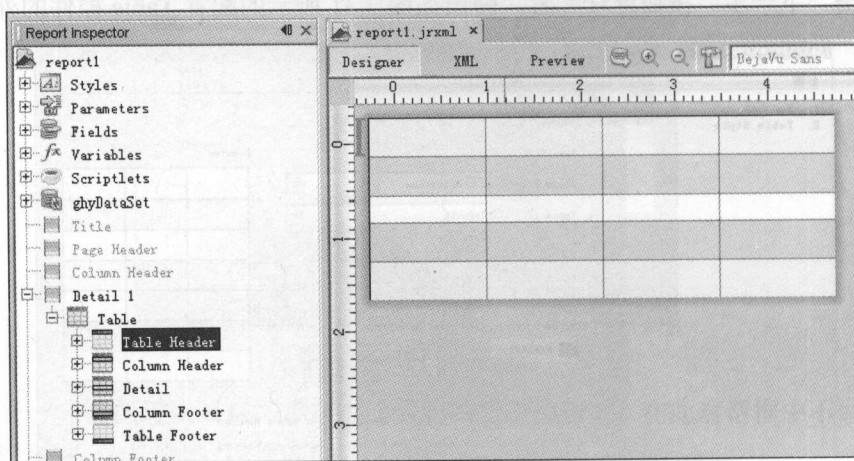


图 7.33 选择 Table Header 行

选择单元格的效果如图 7.34 所示。

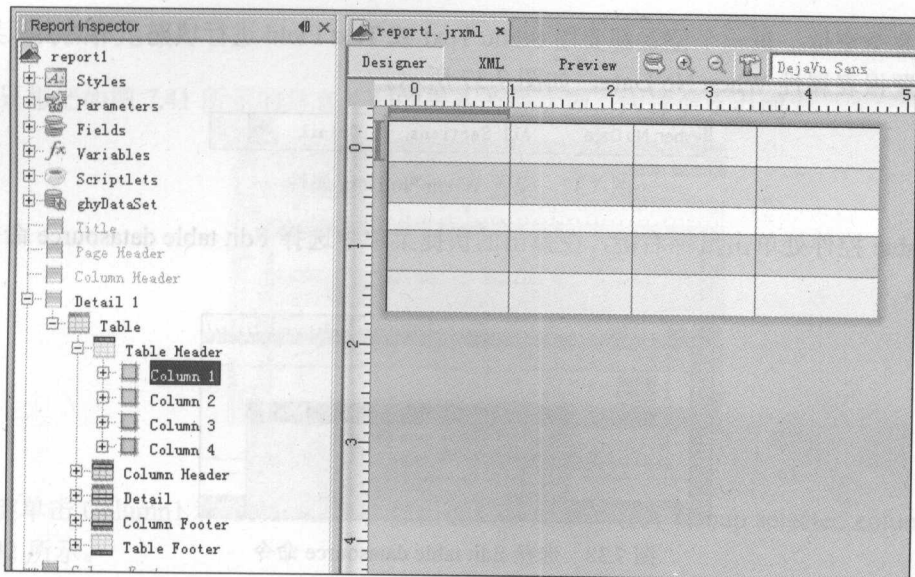


图 7.34 选择单元格的效果图

在显示报表模板界面的下方有两个按钮，用于对报表和 Table 组件设计的切换，如图 7.35 所示。

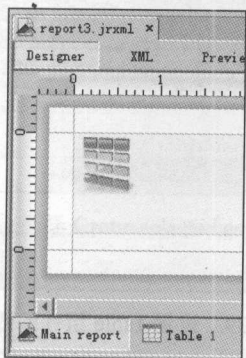


图 7.35 切换的按钮

对 Table 中的内容进行编辑，效果如图 7.36 所示，即添加 4 列和 4 个 Field 对象。

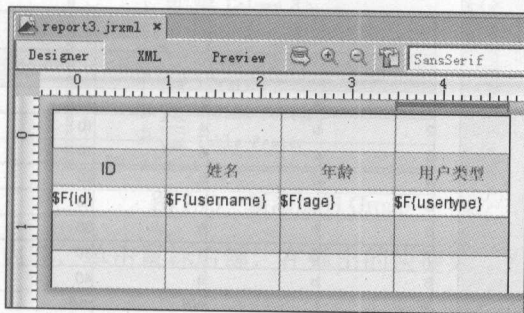


图 7.36 添加 4 列和 4 个 Field 对象

一共 8 个空位，每一个位置都要用 Static Text 或 Text Field 进行填充，不然会报异常。
再设置报表属性 When No Data，如图 7.37 所示。

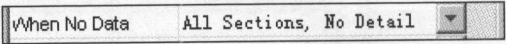


图 7.37 设置 When No Data 属性

在 Table 控件处单击鼠标右键，在弹出的快捷菜单中选择 Edit table datasource 命令，如图 7.38 所示。

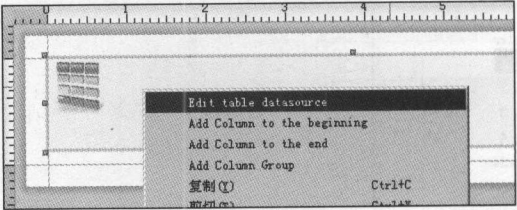


图 7.38 选择 Edit table datasource 命令

设置数据源内容，如图 7.39 所示为设置 Table 组件的数据源。

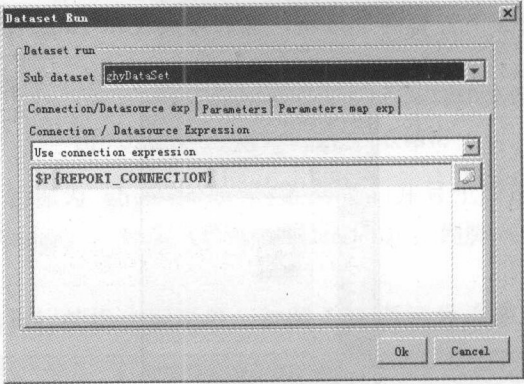


图 7.39 设置 Table 组件的数据源

配置完成后的预览效果如图 7.40 所示。

ID	姓名	年龄	用户类型
1	a	2	10
2	b	4	10
3	c	6	10
4	d	5	20
5	e	7	20
6	f	8	30
7	g	4	30
8	h	9	40
9	i	null	null
10	j	null	null

图 7.40 预览效果

7.3.2 合并单元格

如果想把如图 7.41 所示的具有 4 列的表头改成 2 列，应该怎么办呢？

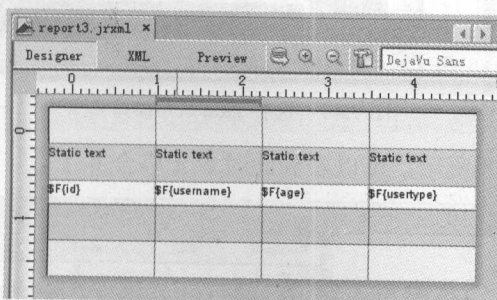


图 7.41 具有 4 列的表头

右键单击 Column1 和 Column2，在弹出的快捷菜单中选择 Group selected columns 命令，如图 7.42 所示。

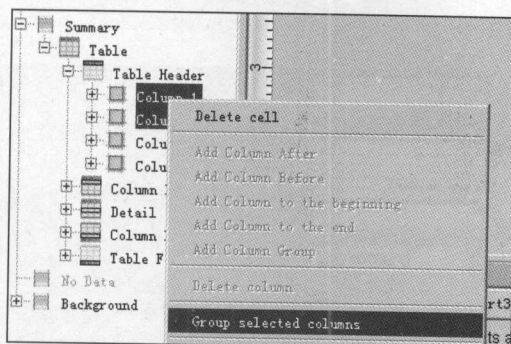


图 7.42 选择 Group selected columns 命令

在 Table Header 中添加了一个 Group，如图 7.43 所示。

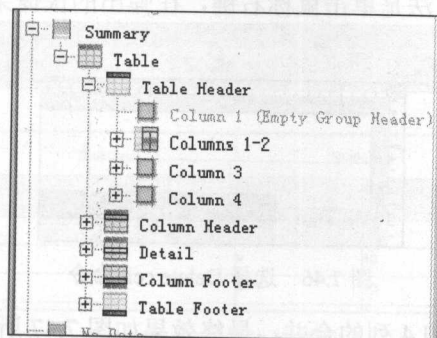


图 7.43 新添加的 Group

但 Group 中并没有内容，单击鼠标右键，在弹出的快捷菜单中选择 Add cell 命令，如图 7.44 所示。

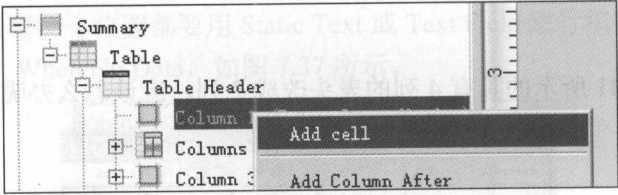


图 7.44 选择 Add cell 命令

此时的表结构如图 7.45 所示，可以看到此时单元格合并了。

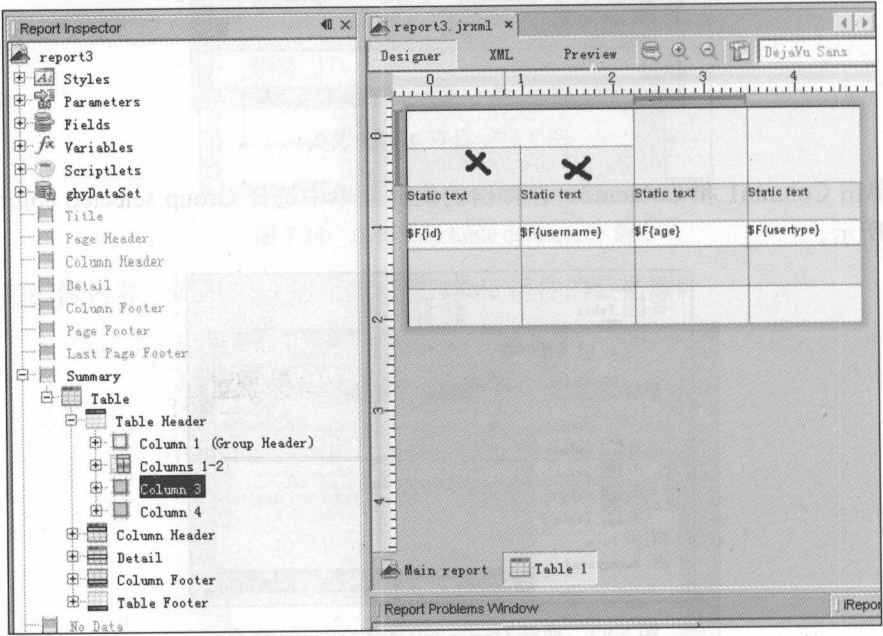




图 7.45 合并单元格

要删除  ，方法是单击鼠标右键，在弹出的快捷菜单中选择 Delete cell 命令，如图 7.46 所示。

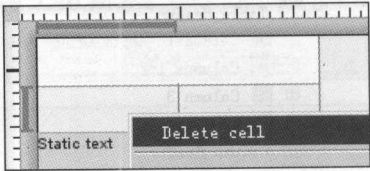


图 7.46 选择 Delete cell 命令

利用相同的方法实现 3 和 4 列的合并，最终效果如图 7.47 所示。

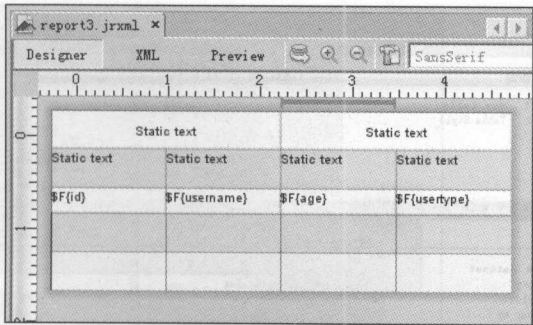


图 7.47 最终效果

还需要设置左上角 cell 中 Static Text 的边框，如图 7.48 所示。

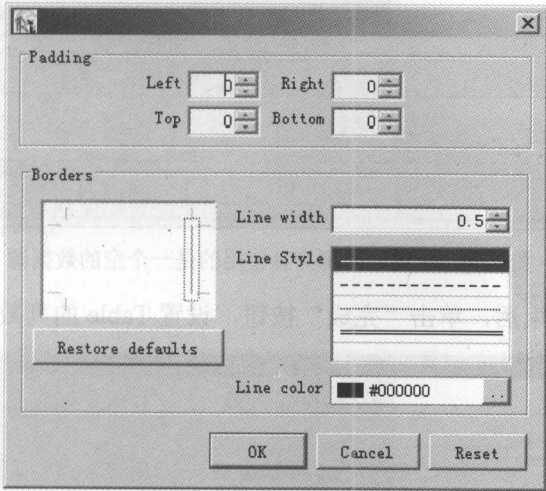


图 7.48 设置边框

报表运行结果如图 7.49 所示。

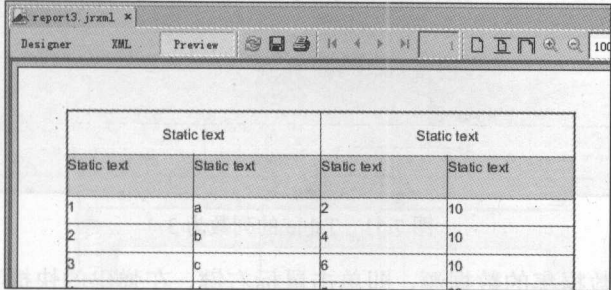


图 7.49 最终效果

7.3.3 使用 JavaBean 作为报表的数据源

在报表模板中添加一个 Table 控件，新建数据源并且使用的是一个空的数据源，如图 7.50 所示。

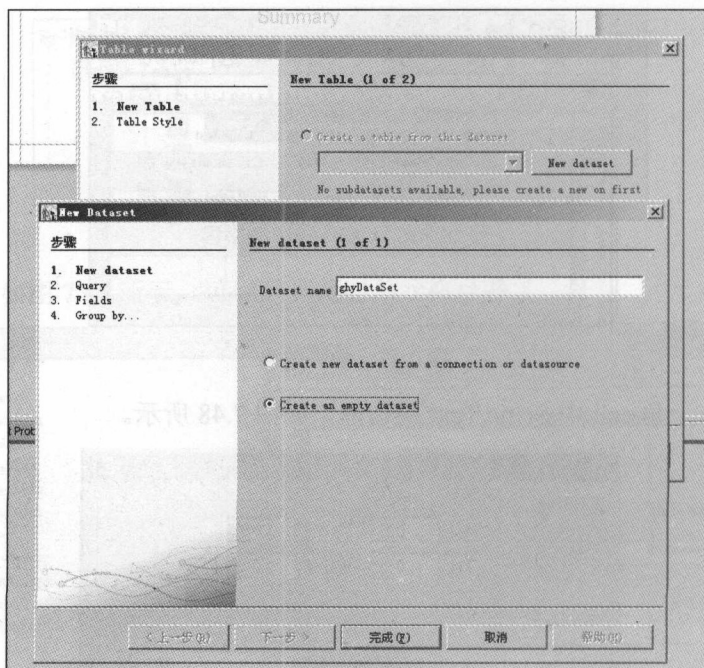


图 7.50 新建数据源并且使用的是一个空的数据源

在图 7.50 中配置结束后，单击“完成”按钮，设置 Table 的列数为 3，如图 7.51 所示。

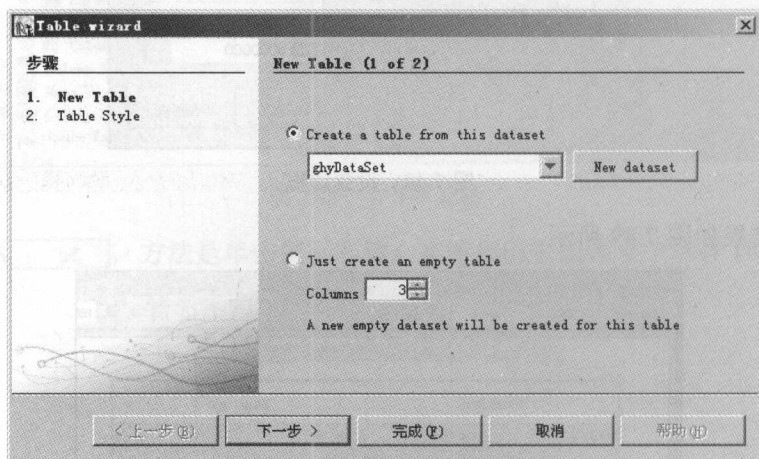


图 7.51 Table 的列数为 3

下面开始配置子数据集的数据源，即单击鼠标右键，在弹出的快捷菜单中选择 Edit Query 命令，如图 7.52 所示。



图 7.52 配置子数据源

设置数据源来自于 JavaBean，如图 7.53 所示。

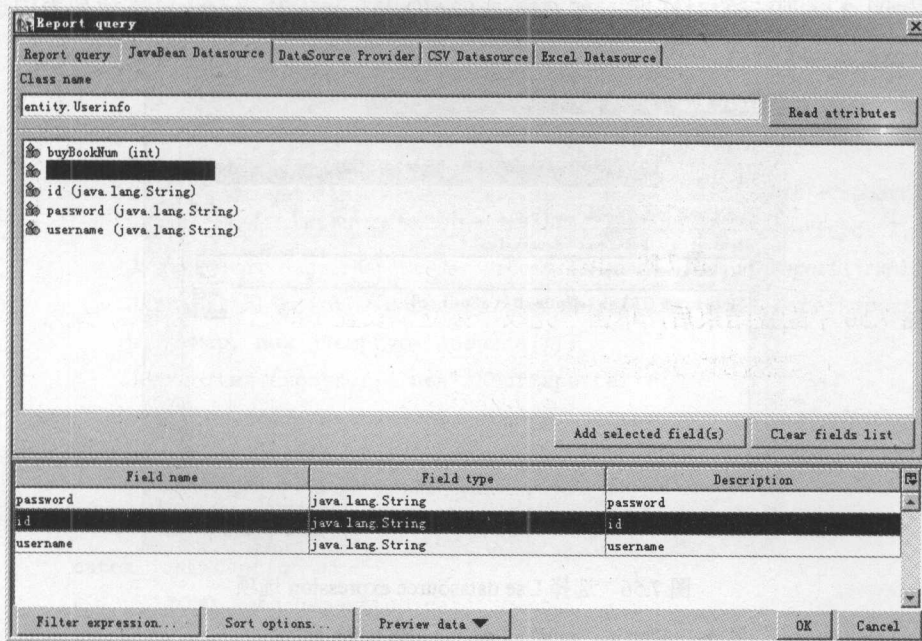


图 7.53 配置 JavaBean 数据源

继续设计报表模板，如图 7.54 所示。

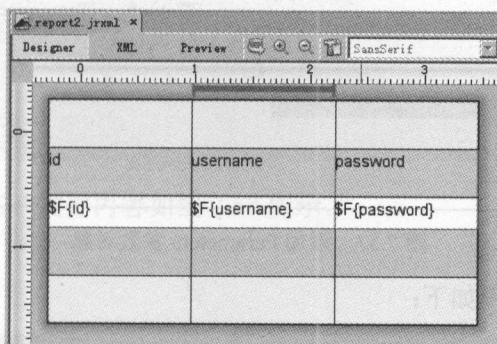


图 7.54 设计报表模板

再编辑 Table 的数据源配置, 即在 Table 控件点上单击鼠标右键, 在弹出的快捷菜单中选择 Edit talde datasource 命令, 如图 7.55 所示。

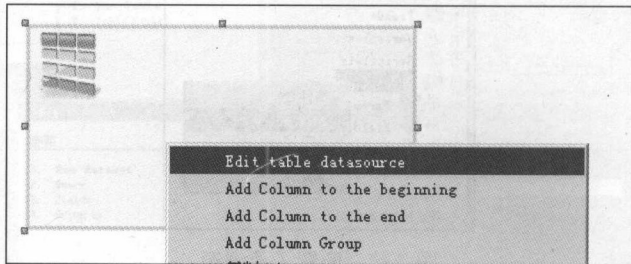


图 7.55 在 Table 控件上单击鼠标右键

弹出如图 7.56 所示的对话框, 在 Connection/Datasource Expression 下拉列表中选择 Use datasource expression 选项。

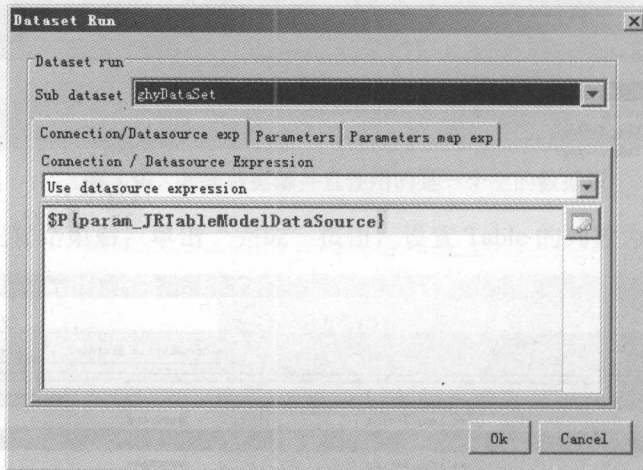


图 7.56 选择 Use datasource expression 选项

在主报表中创建一个 Parameters 参数对象, 如图 7.57 所示。

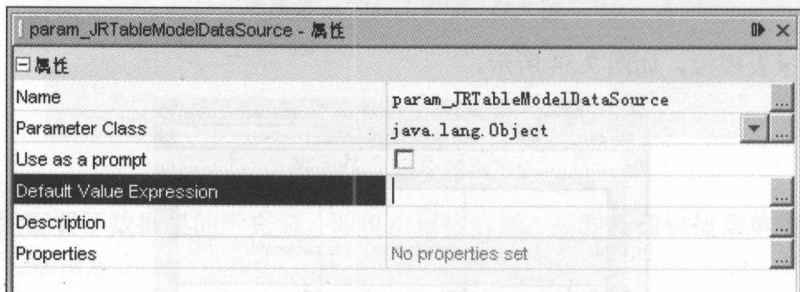


图 7.57 添加 Parameters 参数对象

创建 Servlet, 核心代码如下:

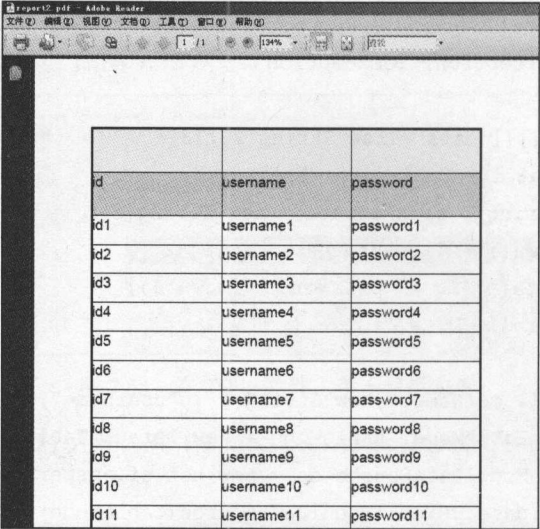
```
public class javaBeanAndTable extends HttpServlet
```

```

{   public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {   try
        {   String[][] data = new String[20][3];
            for (int i = 0; i < 20; i++)
            {   String[] eachRowData = new String[3];
                data[i][0] = "username" + (i + 1);
                data[i][1] = "password" + (i + 1);
                data[i][2] = "id" + (i + 1);
            }
            String[] colName = new String[] { "username", "password", "id" };
            DefaultTableModel tableMode = new DefaultTableModel(data, colName);
            JRTableModelDataSource ds = new net.sf.jasperreports.
            engine.data.JRTableModelDataSource(tableMode);
            Map paramMap = new HashMap();
            paramMap.put("param_JRTableModelDataSource", ds);
            String jrxmlPath=this.getServletContext().getRealPath("/")+"report2.jrxml";
            String pdfPath=this.getServletContext().getRealPath("/")+"report2.pdf";
            JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
            JasperPrint print = JasperFillManager.fillReport(jasperReport,
            paramMap, new JREmptyDataSource());
            JRExporter exporter = new JRPdfExporter();
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, pdfPath);
            exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
            exporter.exportReport();
        }
        catch (JRException e)
        {   // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (SecurityException e)
        {   // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

程序运行后导出的 PDF 文件内容如图 7.58 所示。



The screenshot shows a PDF viewer window titled 'report17.pdf - Adobe Reader'. The window contains a table with 11 rows of data. The first row is a header with columns 'id', 'username', and 'password'. The subsequent rows contain data for users with IDs from 'id1' to 'id11'.

id	username	password
id1	username1	password1
id2	username2	password2
id3	username3	password3
id4	username4	password4
id5	username5	password5
id6	username6	password6
id7	username7	password7
id8	username8	password8
id9	username9	password9
id10	username10	password10
id11	username11	password11

图 7.58 导出的 PDF 文件内容

第8章 实用技巧



导言

本章是对前面一些知识点的总结，也是设计报表时经常遇到的问题，如导出不同文件格式的情况，还有一些在导出时应该注意的事项等，读者应该着重掌握如下内容：

- ★ 不同格式的导出
- ★ 页数的显示及处理
- ★ 一次输出多个报表
- ★ 取消报表分页

8.1 导出各种文件格式

导出报表的功能在软件项目中最为常见，但由于其导出文件的格式种类较多，所以在导出常用的文件格式时总会出现一些问题，本节的主要目的就是演示常用的文件格式导出功能。为了尽可能地全面展示报表，报表中提供了中文、图片、图表等常用的控件。

设计报表模板外观如图 8.1 所示。

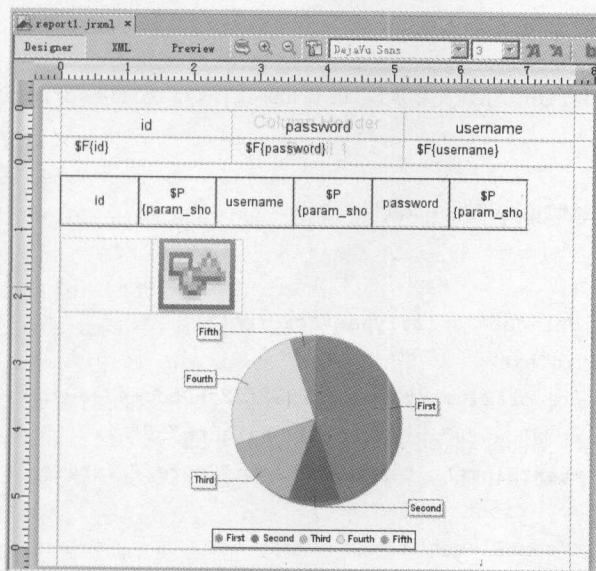


图 8.1 报表模板外观

它的 XML 代码如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<jasperReport xmlns="http://jasperreports.sourceforge.net/jasperreports"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jasperreports.sourceforge.net/jasperreports
    http://jasperreports.sourceforge.net/xsd/jasperreport.xsd"
  name="report1" language="groovy" pageWidth="595" pageHeight="842"
  columnWidth="555" leftMargin="20" rightMargin="20" topMargin="20"
  bottomMargin="20" uuid="bafc3dcb-afd0-498b-8b2c-6ab881da91ed">
  <property name="ireport.zoom" value="0.9090909090909113" />
  <property name="ireport.x" value="0" />
  <property name="ireport.y" value="6" />
  <parameter name="param_showUserInfo" class="java.lang.Object" isForPrompting="false" />
  <parameter name="param_image_src" class="java.lang.String" isForPrompting="false"/>
  <parameter name="param_image_stream" class="java.io.InputStream"
    isForPrompting="false" />
  <queryString><![CDATA[]]> </queryString>
  <field name="buyBookNum" class="java.lang.Integer">
    <fieldDescription><![CDATA[buyBookNum]]></fieldDescription>
  </field>
  <field name="id" class="java.lang.String">
    <fieldDescription><![CDATA[id]]></fieldDescription>
  </field>
  <field name="password" class="java.lang.String">
    <fieldDescription><![CDATA[password]]></fieldDescription>
  </field>
  <field name="username" class="java.lang.String">
    <fieldDescription><![CDATA[username]]></fieldDescription>
  </field>
  <background>
    <band splitType="Stretch" />
  </background>
  <columnHeader>
    <band height="30" splitType="Stretch">
      <staticText>
        <reportElement uuid="f282f72c-0bdc-42e6-a115-87b1b1fe84f0"
          x="0" y="0" width="185" height="30" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
          <font size="16" />
        </textElement>
        <text><![CDATA[id]]></text>
```



```

</staticText>
<staticText>
  <reportElement uuid="81fe56d7-f67f-4d99-a419-b155a0a57eb1"
    x="185" y="0" width="185" height="30" />
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font size="16" />
  </textElement>
  <text><![CDATA[password]]></text>
</staticText>
<staticText>
  <reportElement uuid="d7e7b053-796d-4df9-b7bb-c96ad0829f52"
    x="370" y="0" width="185" height="30" />
  <textElement textAlignment="Center" verticalAlignment="Middle">
    <font size="16" />
  </textElement>
  <text><![CDATA[username]]></text>
</staticText>
</band>
</columnHeader>
<detail>
  <band height="28" splitType="Stretch">
    <textField>
      <reportElement uuid="20b11962-036a-46dc-ad9f-1ac5ceclffbc"
        x="0" y="0" width="185" height="28" />
      <box leftPadding="15" />
      <textElement>
        <font size="14" />
      </textElement>
      <textFieldExpression><![CDATA[${id}]]></textFieldExpression>
    </textField>
    <textField>
      <reportElement uuid="4c9a313d-adc6-4dba-b70c-00542ef7b5df"
        x="185" y="0" width="185" height="28" />
      <box leftPadding="15" />
      <textElement>
        <font size="14" />
      </textElement>
      <textFieldExpression><![CDATA[${password}]]>
    </textFieldExpression>
    </textField>
    <textField>
      <reportElement uuid="f7a983a2-f010-4fc8-92eb-db7bed646bc9"

```

```

        x="370" y="0" width="185" height="28" />
<box leftPadding="15" />
<textElement>
    <font size="14" />
</textElement>
<textFieldExpression><![CDATA[${username}]]>
</textFieldExpression>
</textField>
</band>
</detail>
<summary>
    <band height="416" splitType="Stretch">
        <line>
            <reportElement uuid="ec7b2e27-2432-4fe7-8e63-b16e8f92941b"
                x="168" y="15" width="1" height="52" />
        </line>
        <line>
            <reportElement uuid="451f27a0-63fe-4b41-94b9-852c44122f72"
                x="504" y="15" width="1" height="52" />
        </line>
        <line>
            <reportElement uuid="cfeab39d-86de-4b36-8893-c990fd57f266"
                x="0" y="67" width="505" height="1" />
        </line>
        <line>
            <reportElement uuid="ced340bb-e45c-432b-9b42-5fe63c076a03"
                x="84" y="15" width="1" height="52" />
        </line>
        <line>
            <reportElement uuid="6ac0acf0-518c-46c2-a4ef-30e6b27a3c92"
                x="0" y="14" width="505" height="1" />
        </line>
        <line>
            <reportElement uuid="8dbea711-40d7-4dab-a854-2b8bd0864a92"
                x="420" y="15" width="1" height="52" />
        </line>
        <line>
            <reportElement uuid="f1e9b88b-246e-4770-87fe-a3fe8bd87d63"
                x="0" y="15" width="1" height="52" />
        </line>
        <line>
            <reportElement uuid="514c43cd-6275-42af-9324-50630438695f"

```



```

        x="252" y="15" width="1" height="52" />
    </line>
    <staticText>
        <reportElement uuid="e35e58da-ed64-4bc3-803e-1f549b1f4337"
            x="1" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding
                ="UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <text><![CDATA[id]]></text>
    </staticText>
    <line>
        <reportElement uuid="0995748a-1191-4d4b-82c5-88ef154e7ea3"
            x="336" y="15" width="1" height="52" />
    </line>
    <staticText>
        <reportElement uuid="e35e58da-ed64-4bc3-803e-1f549b1f4337"
            x="169" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding=
                "UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <text><![CDATA[username]]></text>
    </staticText>
    <staticText>
        <reportElement uuid="e35e58da-ed64-4bc3-803e-1f549b1f4337"
            x="337" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding
                ="UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <text><![CDATA[password]]></text>
    </staticText>
    <textField>
        <reportElement uuid="0b43d687-4b0c-44f4-8a9f-e320b8b26fdc"
            x="85" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding
                ="UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <textFieldExpression>
            <![CDATA[${P{param_showUserInfo}.getId()}]>

```



```

        </textFieldExpression>
    </textField>
    <textField>
        <reportElement uuid="0b43d687-4b0c-44f4-8a9f-e320b8b26fdc"
            x="253" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding
                ="UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <textFieldExpression>
            <![CDATA[${P{param_showUserInfo}.getUsername()}]>
        </textFieldExpression>
    </textField>
    <textField>
        <reportElement uuid="0b43d687-4b0c-44f4-8a9f-e320b8b26fdc"
            x="421" y="15" width="83" height="51" />
        <textElement textAlignment="Center" verticalAlignment="Middle">
            <font size="13" pdfFontName="STSong-Light" pdfEncoding
                ="UniGB-UCS2-H" isPdfEmbedded="true" />
        </textElement>
        <textFieldExpression>
            <![CDATA[${P{param_showUserInfo}.getPassword()}]>
        </textFieldExpression>
    </textField>
    <image hAlign="Center" vAlign="Middle">
        <reportElement uuid="934ebbef-8040-42e9-a02e-da7cd418df6d"
            x="1" y="80" width="99" height="82" />
        <imageExpression>
            <![CDATA[${P{param_image_src}}]></imageExpression>
    </image>
    <image hAlign="Center" vAlign="Middle">
        <reportElement uuid="1d3a0df7-1c5c-4a65-8792-a0df46e7827f"
            x="100" y="80" width="99" height="82" />
        <imageExpression>
            <![CDATA[${P{param_image_stream}}]></imageExpression>
    </image>
    <pieChart>
        <chart>
            <reportElement uuid="07b69ff1-766e-4d63-a7b9-69a5ba525aa1"
                x="1" y="162" width="554" height="254" />
            <chartTitle />
            <chartSubtitle />
        </chart>
    </pieChart>

```

```

        <chartLegend />
    </chart>
    <pieDataset>
        <keyExpression><![CDATA[${F{id}}]></keyExpression>
        <valueExpression><![CDATA[${F{buyBookNum}}]></valueExpression>
        <labelExpression><![CDATA[${F{username}}]></labelExpression>
    </pieDataset>
    <piePlot><plot/><itemLabel />
</piePlot>
</pieChart>
</band>
</summary>
</jasperReport>

```

8.1.1 导出.xls 文件

创建导出 Excel 文件的 Servlet，示例代码如下：

```

public class excel extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
            String xlsPath = this.getServletContext().getRealPath("/")+"report1.xls";
            JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
            // 获得父类型的 leftMargin 属性进行反射，主要的目的是
            // 导出 excel 文件时取消左上下的 margin 边距
            Field margin = jasperReport.getClass().getSuperclass()
                .getDeclaredField("leftMargin");

```



```

        margin.setAccessible(true);
        margin.setInt(jasperReport, 0);
        margin = jasperReport.getClass().getSuperclass().getDeclaredField(
            "topMargin");
        margin.setAccessible(true);
        margin.setInt(jasperReport, 0);
        margin = jasperReport.getClass().getSuperclass().getDeclaredField(
            "bottomMargin");
        margin.setAccessible(true);
        margin.setInt(jasperReport, 0);
        JasperPrint print = JasperFillManager.fillReport(jasperReport,
            paramMap, new JRBeanCollectionDataSource(listUserinfo));
        JRExporter exporter = new JRXlsExporter();
        // 取消导出 excel 时超大数字格式的字符串会自动转成科学计数法进行显示
        exporter.setParameter(JRXlsExporterParameter.IS_DETECT_CELL_TYPE,
            Boolean.TRUE);
        exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, xlsPath);
        exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
        exporter.setParameter(
            JRXlsExporterParameter.IS_REMOVE_EMPTY_SPACE_BETWEEN_ROWS, Boolean.TRUE);
        // 是否删除空行
        exporter.setParameter(JRXlsExporterParameter.IS_ONE_PAGE_PER_SHEET,
            Boolean.TRUE); // 是否把每一页放入一个新的 sheet 中
        exporter.setParameter(JRXlsExporterParameter.IS_WHITE_PAGE_BACKGROUND,
            Boolean.FALSE); // 背景是否白色
        exporter.exportReport();
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (SecurityException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (NoSuchFieldException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (IllegalArgumentException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

```



```

    }
    catch (IllegalAccessException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

程序运行后导出.xls 文件，第 1 页的内容如图 8.2 所示。

	A	B	C	D
	id	password	username	
1	id1	password1	username1	
2	id2	password2	username2	
3	id3	password3	username3	
4	id4	password4	username4	
5	id5	password5	username5	
6	id6	password6	username6	
7	id7	password7	username7	
8	id8	password8	username8	
9	id9	password9	username9	
10	id10	password10	username10	
11	id11	password11	username11	
12	id12	password12	username12	
13	id13	password13	username13	
14	id14	password14	username14	
15	id15	password15	username15	
16	id16	password16	username16	
17	id17	password17	username17	
18	id18	password18	username18	
19	id19	password19	username19	
20	id20	password20	username20	
21				
22				
23				

图 8.2 第 1 页内容

第 2 页的内容如图 8.3 所示。

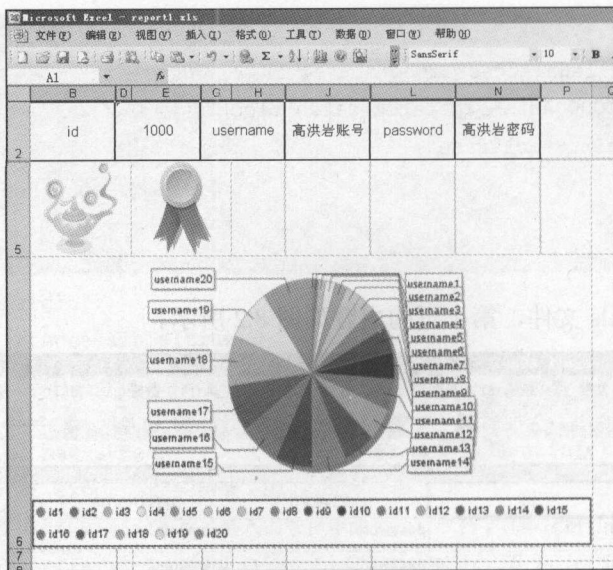


图 8.3 第 2 页内容

从导出.xls 文件的结果来看，一切正常！

若想导出.xls 文件并下载，该如何处理呢？

下面要实现的效果是将报表导出成.xls 文件后下载，核心代码如下：

```
public class exportAndDownload extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath = this.getServletContext().getRealPath("/") + "report1.jrxml";
            String pdfPath = this.getServletContext().getRealPath("/") + "report1.pdf";
            JasperReport jasperReport = JasperCompileManager.compileReport(jrxmlPath);
```

```

JasperPrint print = JasperFillManager.fillReport(jasperReport,
paramMap, new JRBeanCollectionDataSource(listUserInfo));
JRExporter exporter = new JRXlsExporter();
response.setContentType("application/vnd.ms-excel");
response.setHeader("Content-Disposition", "attachment;filename=\"\"
+ java.net.URLEncoder.encode("高洪岩文件.xls", "utf-8") + "\"");
exporter.setParameter(JRXlsExporterParameter.
IS_REMOVE_EMPTY_SPACE_BETWEEN_ROWS, Boolean.TRUE);
exporter.setParameter(JRXlsExporterParameter.IS_ONE_PAGE_PER_SHEET,
Boolean.FALSE);
exporter.setParameter(JRXlsExporterParameter.IS_WHITE_PAGE_BACKGROUND,
Boolean.FALSE);
exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
exporter.setParameter(JRExporterParameter.CHARACTER_ENCODING, "utf-8");
exporter.setParameter(JRExporterParameter.OUTPUT_STREAM, response
.getOutputStream());
exporter.exportReport();
}
catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (SecurityException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

程序运行后的效果如图 8.4 所示。

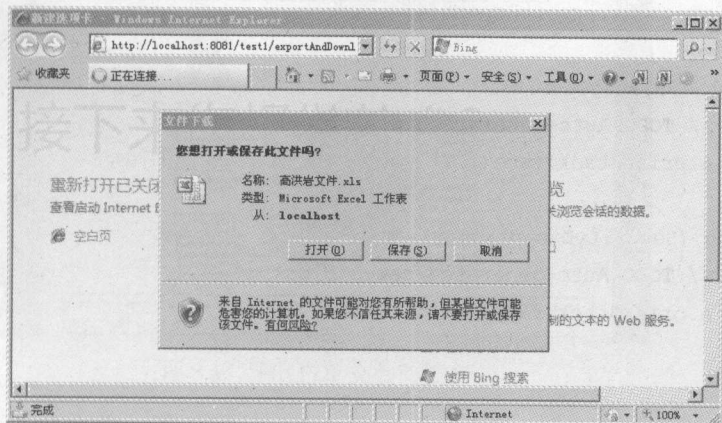


图 8.4 运行效果

8.1.2 导出 PDF 文件

创建导出 PDF 的 Servlet，核心代码如下：

```
public class pdf extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
            String pdfPath=this.getServletContext().getRealPath("/")+"report1.pdf";
            JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
            JasperPrint print = JasperFillManager.fillReport(jasperReport,
                paramMap, new JRBeanCollectionDataSource(listUserinfo));
            JRExporter exporter = new JRPdfExporter();
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, pdfPath);
            exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
            exporter.exportReport();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (SecurityException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```

程序运行后导出的 PDF 文件内容如图 8.5 所示。

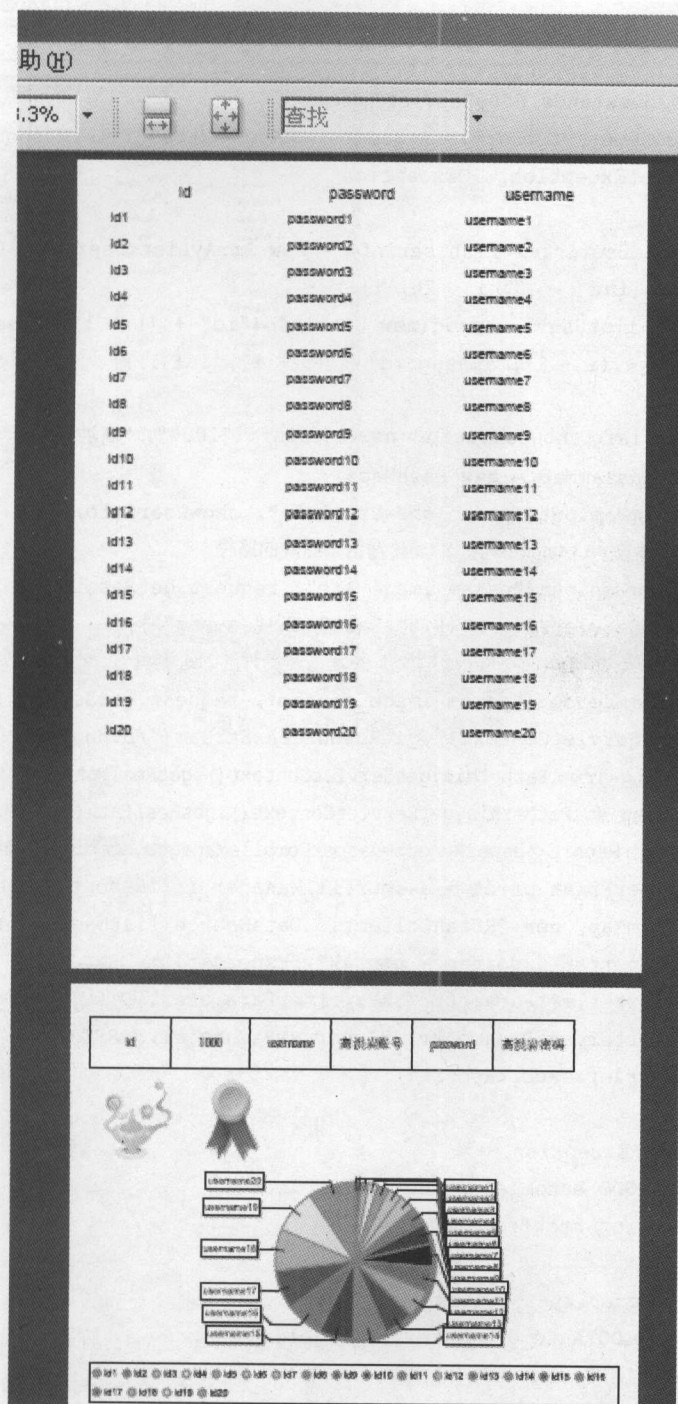


图 8.5 导出 PDF 文件的内容

可以看到导出的 PDF 文件一切正常。

8.1.3 导出 DOC 文件

创建导出 DOC 的 Servlet 对象，核心代码如下：

```
public class word extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
            String docPath=this.getServletContext().getRealPath("/")+"report1.doc";
            JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
            JasperPrint print = JasperFillManager.fillReport(jasperReport,
                paramMap, new JRBeanCollectionDataSource(listUserinfo));
            JRExporter exporter = new JRRtfExporter();
            exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME, docPath);
            exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
            exporter.exportReport();
        }
        catch (JRException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (SecurityException e)
        {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```


导出的 DOC 文档内容如图 8.6 所示。

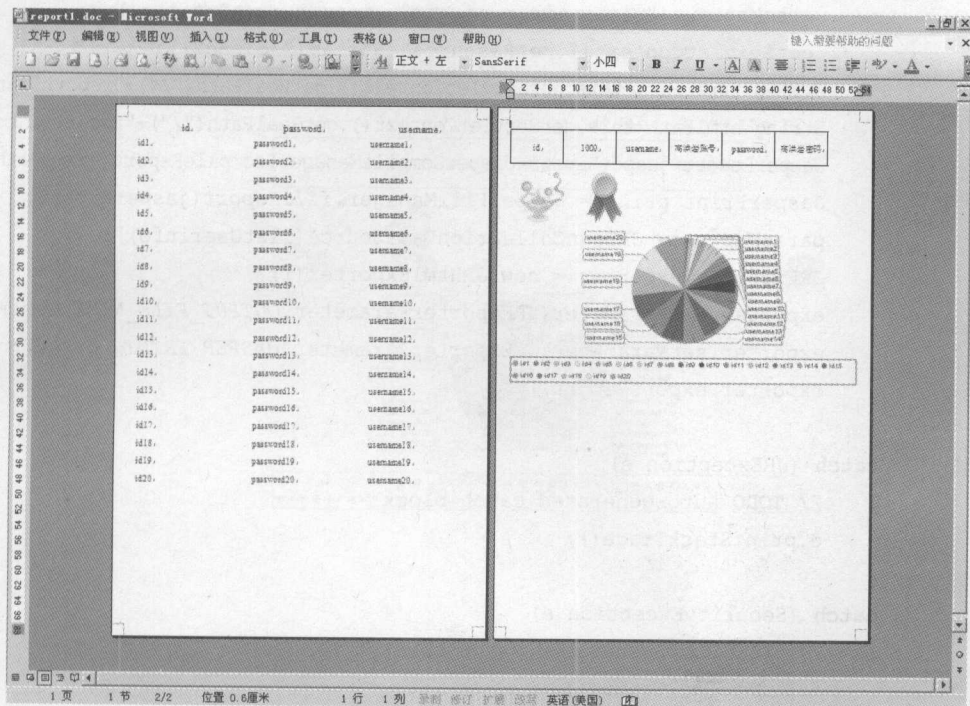


图 8.6 导出的 DOC 文档内容

8.1.4 导出 HTML 文件

1. 导出 HTML 文件的操作

创建导出 HTML 的 Servlet，核心代码如下：

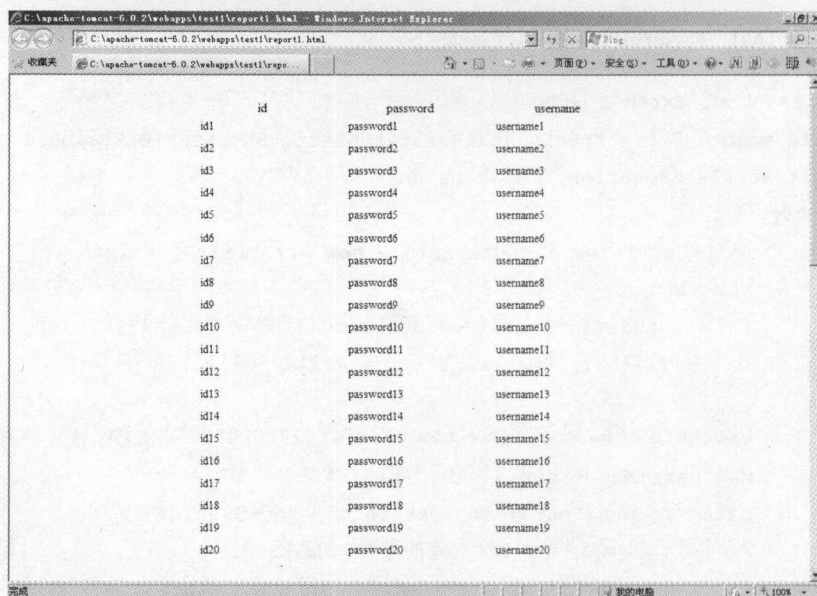
```
public class html extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
        }
    }
}
```

```

// 参数 param_image_stream 传递的是图片的二进制流
paramMap.put("param_image_stream", request.getSession()
    .getServletContext().getResourceAsStream("/b.png"));
String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
String htmlPath=this.getServletContext().getRealPath("/")+"report1.html";
JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
JasperPrint print = JasperFillManager.fillReport(jasperReport,
    paramMap, new JRBeanCollectionDataSource(listUserInfo));
JRExporter exporter = new JRHtmlExporter();
exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,htmlPath);
exporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
exporter.exportReport();
}
catch (JRException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
catch (SecurityException e)
{
    // TODO Auto-generated catch block
    e.printStackTrace();
}
}
}

```

导出的 HTML 文件部分内容 1 如图 8.7 所示。



id	password	username
id1	password1	username1
id2	password2	username2
id3	password3	username3
id4	password4	username4
id5	password5	username5
id6	password6	username6
id7	password7	username7
id8	password8	username8
id9	password9	username9
id10	password10	username10
id11	password11	username11
id12	password12	username12
id13	password13	username13
id14	password14	username14
id15	password15	username15
id16	password16	username16
id17	password17	username17
id18	password18	username18
id19	password19	username19
id20	password20	username20

图 8.7 HTML 文件部分内容 1

HTML 文件部分内容 2 如图 8.8 所示。

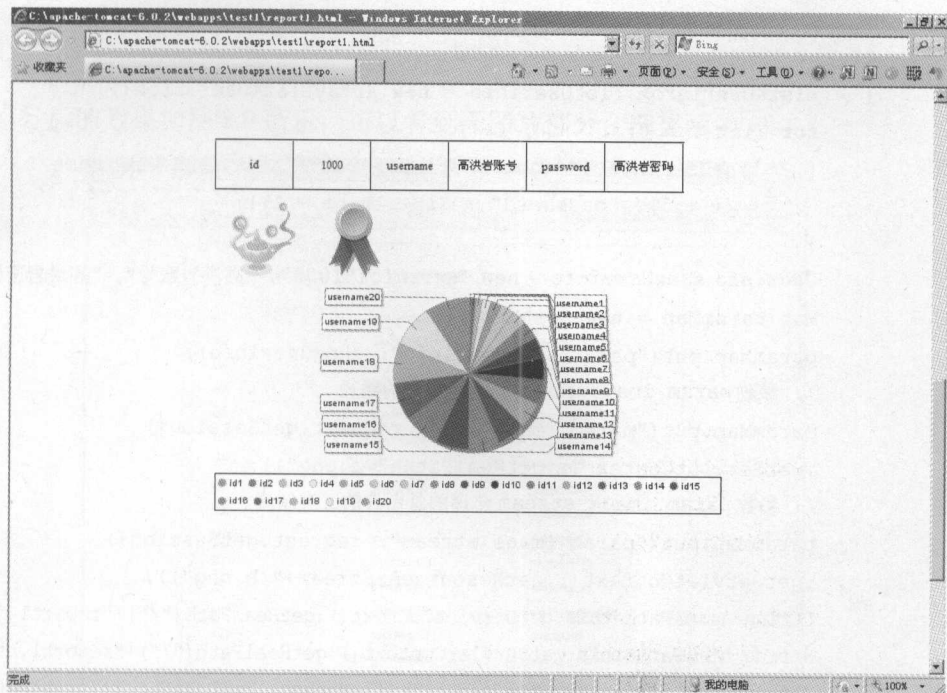


图 8.8 HTML 文件部分内容 2

还可以加入如下参数。

- 设置在 HTML 中打印时分割分页的 HTML 代码:

```
exporter.setParameter(JRHtmlExporterParameter.BETWEEN_PAGES_HTML, "值为 html 代码");
```

- 指定打印第几页:

```
exporter.setParameter(JRHtmlExporterParameter.PAGE_INDEX, page);
```

- 把报表中的图片解析到指定的文件夹中:

```
exporter.setParameter(JRHtmlExporterParameter.IMAGES_DIR_NAME, request.getSession().getServletContext().getRealPath("exportImage"));
```

- 是否把报表中的图片解析出来:

```
exporter.setParameter(JRHtmlExporterParameter.IS_OUTPUT_IMAGES_TO_DIR, Boolean.TRUE);
```

2. 在 IE 上显示 HTML 内容

下面将要演示将 HTML 的内容在 IE 上显示的效果, Servlet 核心代码如下:


```
public class htmlShowIE extends HttpServlet
```



```

{   public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException
    {   try
        {   List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {   listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
            String htmlPath=this.getServletContext().getRealPath("/")+"report1.html";
            JasperReport jasperReport=JasperCompileManager.compileReport(jrxmlPath);
            JasperPrint print = JasperFillManager.fillReport(jasperReport,
                paramMap, new JRBeanCollectionDataSource(listUserinfo));
            // 设置格式
            response.setContentType("text/html");
            PrintWriter printWriter = response.getWriter();
            JRHtmlExporter htmlExporter = new JRHtmlExporter();
            // request.getSession().setAttribute(ImageServlet.
            // DEFAULT_JASPER_PRINT_SESSION_ATTRIBUTE,jasperPrint);
            htmlExporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
            htmlExporter.setParameter(JRExporterParameter.OUTPUT_WRITER,printWriter);
            // htmlExporter.setParameter(JRHtmlExporterParameter.IMAGES_URI,
            // "image?image=");
            htmlExporter.exportReport();
            printWriter.close();
        }
        catch (JRException e)
        {   // TODO Auto-generated catch block
            e.printStackTrace();
        }
        catch (SecurityException e)
        {   // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}

```

程序运行后的效果如图 8.9 所示，可以看到无图片部分全部显示为.

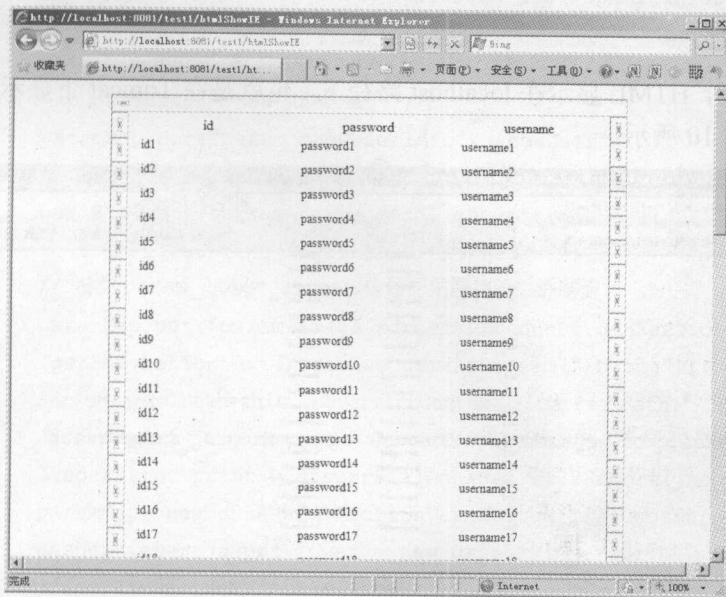


图 8.9 无图片显示红叉

出错的原因是没有配置 Image 的 URI 参数，更改核心代码如下：

```
response.setContentType("text/html");
response.setCharacterEncoding("utf-8");
PrintWriter printWriter = response.getWriter();
JRHtmlExporter htmlExporter = new JRHtmlExporter();
htmlExporter.setParameter(JRExporterParameter.JASPER_PRINT, print);
htmlExporter.setParameter(JRExporterParameter.OUTPUT_WRITER,
    printWriter);
request.getSession().setAttribute(ImageServlet.DEFAULT_JASPER_PRINT_SESSION_ATTRIBUTE, print);
htmlExporter.setParameter(JRHtmlExporterParameter.IMAGES_URI, "image?image=");
htmlExporter.exportReport();
printWriter.close();
```

然后在 web.xml 中配置路径为 image 的 Servlet，因为 JasperReport 要使用此 Servlet 来生成图片流。

在文件 web.xml 中添加配置如下：

```
<servlet>
    <servlet-name>ImageServlet</servlet-name>
```

```

<servlet-class>net.sf.jasperreports.j2ee.servlets.ImageServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>ImageServlet</servlet-name>
    <url-pattern>/image</url-pattern>
</servlet-mapping>

```

再次运行项目，HTML 显示在 localhost 路径下，也就是在 Tomcat 下显示出了报表，正确显示的报表如图 8.10 所示。

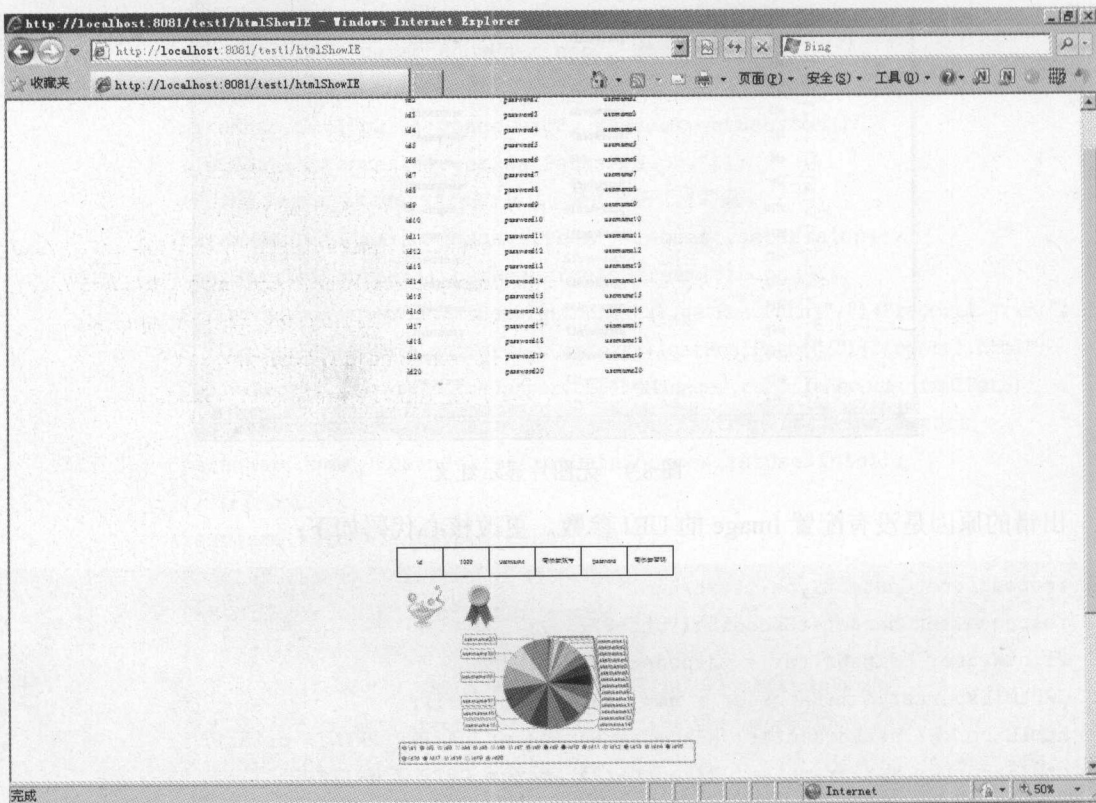


图 8.10 正确显示的报表

如果出现图片缓存的问题，可以在 URL 中加入随机数即可，代码如下：

```

htmlExporter.setParameter(JRHtmlExporterParameter.IMAGES_URI,
    "image?" + "time=" + (new Date()).getTime() + "&image=");

```

3. 在 IE 中使用 JasperViewer 对象预览报表

创建核心 Servlet 代码如下：

```

public class useJasperViewer extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException

```



```

{
    try
    {
        List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
        for (int i = 0; i < 20; i++)
        {
            listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                + (i + 1), "password" + (i + 1), i + 1));
        }
        Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
        Map paramMap = new HashMap();
        paramMap.put("param_showUserinfo", showUserinfo);
        // 参数 param_image_src 传递的是图片的路径
        paramMap.put("param_image_src", request.getSession()
            .getServletContext().getRealPath("/a.png"));
        // 参数 param_image_stream 传递的是图片的二进制流
        paramMap.put("param_image_stream", request.getSession()
            .getServletContext().getResourceAsStream("/b.png"));
        String jrxmlPath=this.getServletContext().getRealPath("/")+"report1.jrxml";
        JasperReport jasperReport = JasperCompileManager.compileReport(jrxmlPath);
        JasperPrint print = JasperFillManager.fillReport(jasperReport,
            paramMap, new JRBeanCollectionDataSource(listUserinfo));
        JasperViewer jasperViewer = new JasperViewer(print, false);
        jasperViewer.setVisible(true);
    }
    catch (JRException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    catch (SecurityException e)
    {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
}

```

在 IE 中输入 URL 网址后弹出 JasperViewer 窗口，如图 8.11 所示。

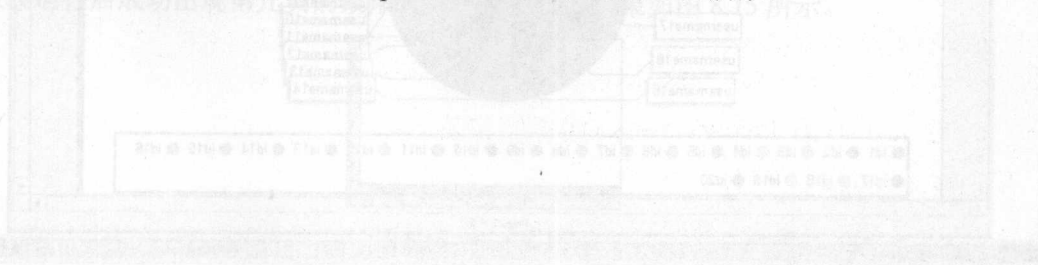


图 8.11 在 IE 中输入 URL 网址后弹出 JasperViewer 窗口

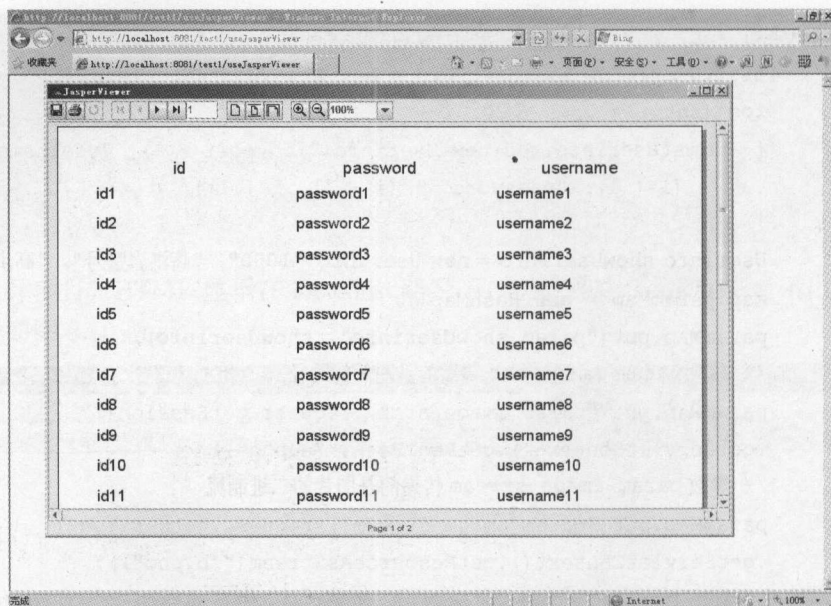


图 8.11 弹出 JasperViewer 窗口

可以看到，系统已支持图片和图表的显示，如图 8.12 所示。

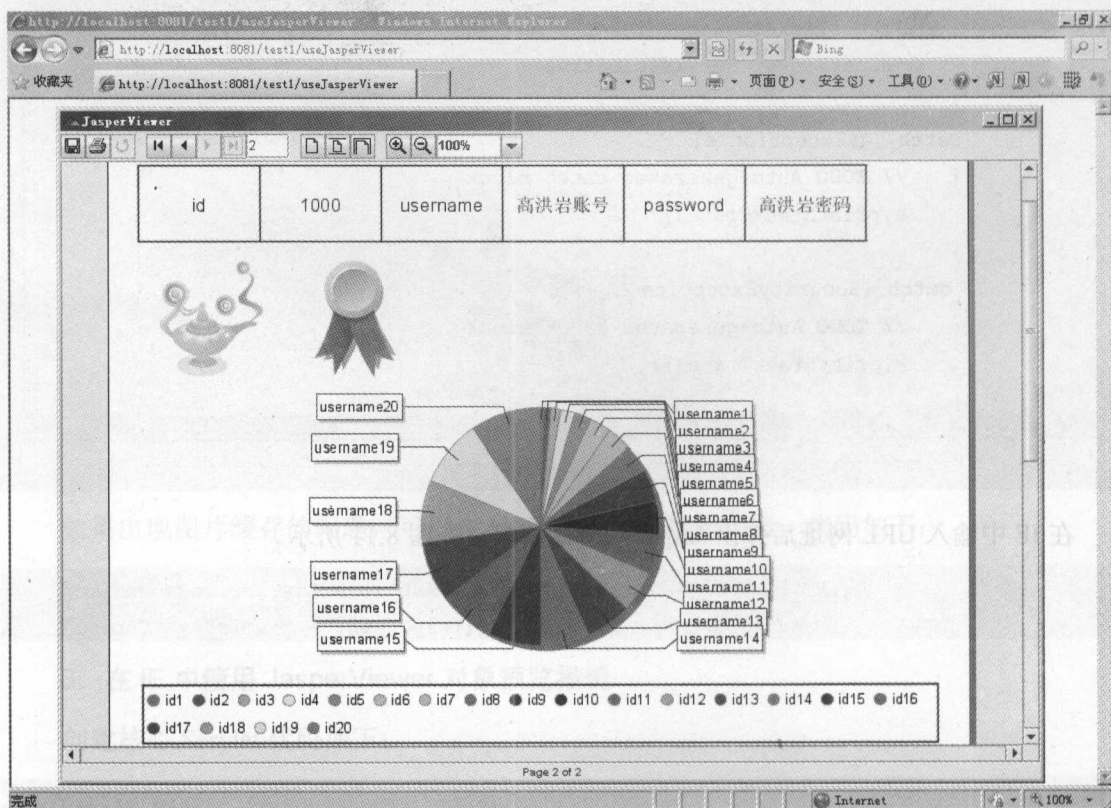


图 8.12 支持图片和图表显示

8.2 取消报表分页

取消报表分页的功能主要应用在到电信部门打印电话单、到超市结账时打印的凭证条等,这种类型的打印不需要分页。打印出来的内容呈长条状,那在 iReport 中如何取消分页呢?其实很简单,设置 Ignore pagination 属性值为 true 即可,如图 8.13 所示。

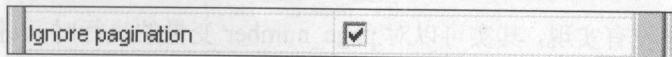


图 8.13 取消分页属性

程序运行后不再出现分页。

8.3 实现当前页/总页数的效果

iReport 中有一个默认变量 page_number, 使用该变量可以实现当前页/总页数的效果。将变量 page_number 放入报表模板的页脚中, 如图 8.14 所示。

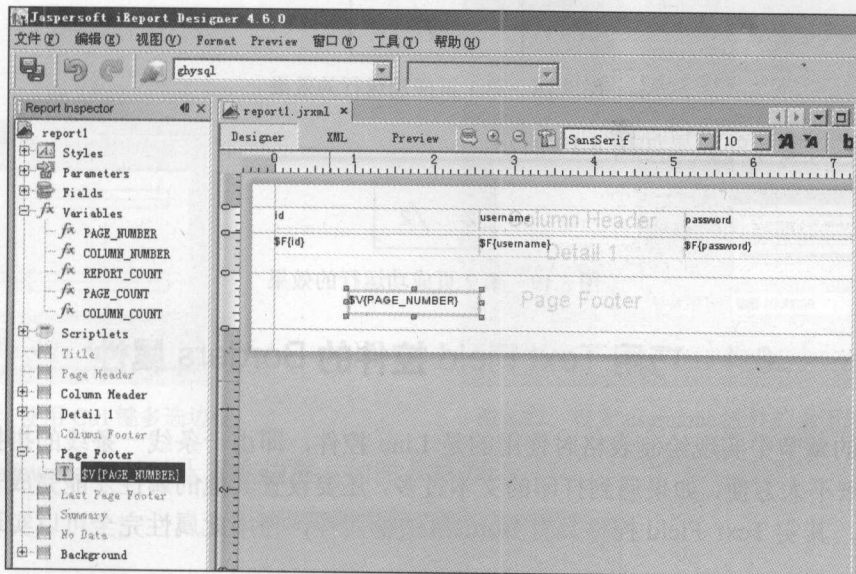


图 8.14 将 page_number 变量放在页脚

报表运行后成功出现第几页的效果, 第 1 页的页码效果如图 8.15 所示。

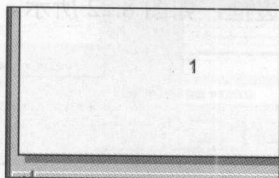


图 8.15 第 1 页页码

第 2 页的页码也显示出来了, 效果如图 8.16 所示。

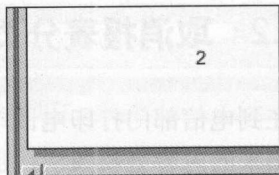
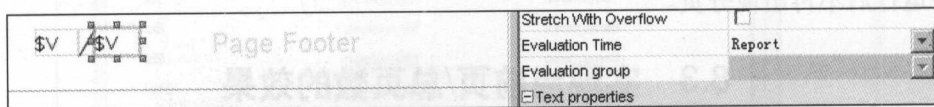


图 8.16 第 2 页页码

但页数的效果并没有实现，其实可以对 `page_number` 变量的执行时间进行设置，再添加一个 `page_number` 变量，并且设置第 2 个 `page_number` 变量的 `Evaluation Time` 属性值为 `Report`，如图 8.17 所示。

图 8.17 设置第 2 个 `page_number` 变量的 `Evaluation Time` 属性值为 `Report`

再次运行报表，第 1 页成功运行的效果如图 8.18 所示。

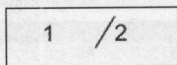


图 8.18 第 1 页成功运行的效果

第 2 页成功运行的效果如图 8.19 所示。

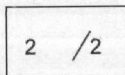


图 8.19 第 2 页成功运行的效果

8.4 巧用 Text Field 控件的 Borders 属性

在前面的章节中实现绘制表格时使用的是 `Line` 控件，即由一条线一条线的拼接成一个表格，使用起来不太方便，如果遇到打印的文本过多，还要设置其他的属性才能完成长度自适应增长的效果，其实 `Text Field` 控件具有 `Borders` 边框属性，使用此属性完全可以实现表格的效果。

设置最左边 `id` 的 `Text Field` 控件的边框，例如设置 `id` 控件的左上下边框如图 8.20 所示。也可以使用 `Ctrl` 键在图 8.21 中进行边框的多选，从而一起设置 `Line width` 属性。继续设置 `username` 控件的上下边框，如图 8.22 所示。

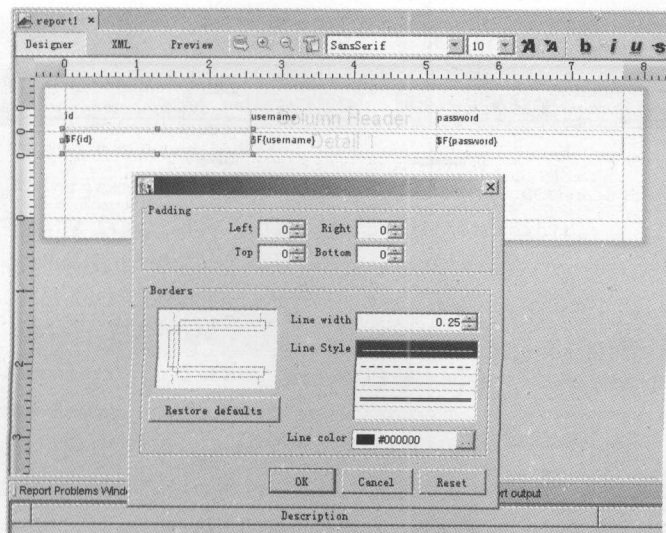


图 8.20 设置 id 控件的左上下边框

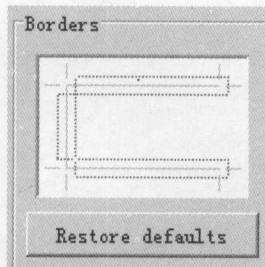


图 8.21 使用 Ctrl 键多选边框

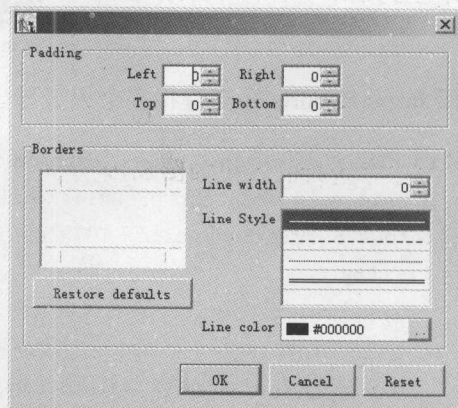


图 8.22 设置 username 控件的上下边框

设置 password 控件的上下右边框，如图 8.23 所示。

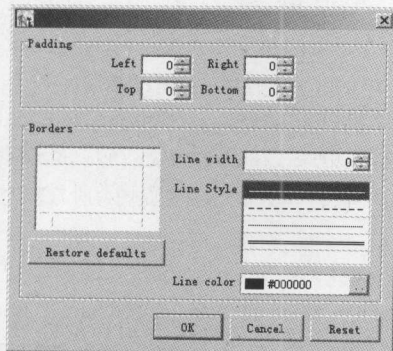


图 8.23 设置 password 边框的上下右边框

运行报表，可以看到通过上面的操作即可简单方便地成功实现带边框的表格，如图 8.24

所示。

id	username	password
1	a	b
2	a	b
3	a	b
4	a	b
5	a	b
6	a	b
7	a	b
8	a	b
9	a	b
10	a	b

图 8.24 实现带边框的表格

8.5 一次输出多个报表

框架 JasperReports 还支持将多个报表导出成一个 PDF 文件, 创建核心 Servlet, 代码如下:

```
public class moreReportExport extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException
    {
        try
        {
            List<Userinfo> listUserinfo = new ArrayList<Userinfo>();
            for (int i = 0; i < 20; i++)
            {
                listUserinfo.add(new Userinfo("id" + (i + 1), "username"
                    + (i + 1), "password" + (i + 1), i + 1));
            }
            Userinfo showUserinfo = new Userinfo("1000", "高洪岩账号", "高洪岩密码", 0);
            Map paramMap = new HashMap();
            paramMap.put("param_showUserinfo", showUserinfo);
            // 参数 param_image_src 传递的是图片的路径
            paramMap.put("param_image_src", request.getSession()
                .getServletContext().getRealPath("/a.png"));
            // 参数 param_image_stream 传递的是图片的二进制流
            paramMap.put("param_image_stream", request.getSession()
                .getServletContext().getResourceAsStream("/b.png"));
            String jrxmlPath = this.getServletContext().getRealPath("/") + "report1.jrxml";
            JasperReport jasperReport = JasperCompileManager.compileReport(jrxmlPath);
            JasperPrint print = JasperFillManager.fillReport(jasperReport,
                paramMap, new JRBeanCollectionDataSource(listUserinfo));
            // ///////////////////////////////////
        }
    }
}
```



```

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");
String url = "jdbc:sqlserver://localhost:1079;databaseName=ghydb";
String username = "sa";
String password = "";
String jrxml2Path = this.getServletContext().getRealPath("/")+"report2.jrxml";
String finalPDFPath=this.getServletContext().
getRealPath("/")+"doubleReport.pdf";
Connection connection=DriverManager.getConnection(url,username,password);
JasperReport jasperReport2 = JasperCompileManager.compileReport(jrxml2Path);
JasperPrint print2 = JasperFillManager.fillReport(jasperReport2,
new HashMap(), connection);
List printList = new ArrayList();
printList.add(print);
printList.add(print2);
JRExporter exporter = new JRPdfExporter();
exporter.setParameter(JRExporterParameter.OUTPUT_FILE_NAME,finalPDFPath);
exporter.setParameter(JRExporterParameter.JASPER_PRINT_LIST,printList);
exporter.exportReport();
}
catch (JRException e)
{ // TODO Auto-generated catch block
e.printStackTrace();
}
catch (SecurityException e)
{ // TODO Auto-generated catch block
e.printStackTrace();
}
catch (ClassNotFoundException e)
{ // TODO Auto-generated catch block
e.printStackTrace();
}
catch (SQLException e)
{ // TODO Auto-generated catch block
e.printStackTrace();
}
}
}

```

运行程序的结果如图 8.25 所示，可以看到多个报表已合并成一个 PDF 文件。

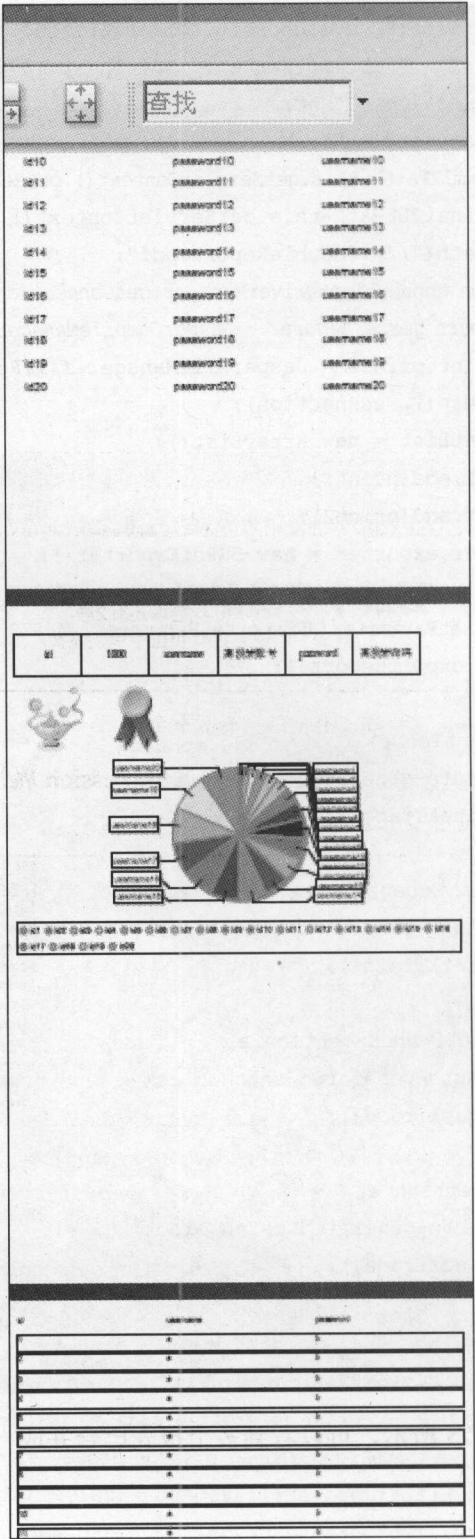


图 8.25 多个报表合并成一个 PDF 文件

8.6 静态文本多行显示

若遇到 Static Text 控件中的文本过多并且不自动换行,默认高度还没有随着内容的多少而自动增长时,效果如图 8.26 所示。预览后文字显示不全,效果如图 8.27 所示,这种静态文本过多的情况应该怎么解决呢?其实很简单,利用变量对象 Variables 转换一下就可以了。

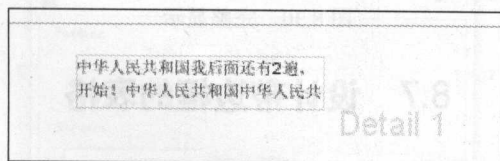


图 8.26 不换行的 Static Text 控件

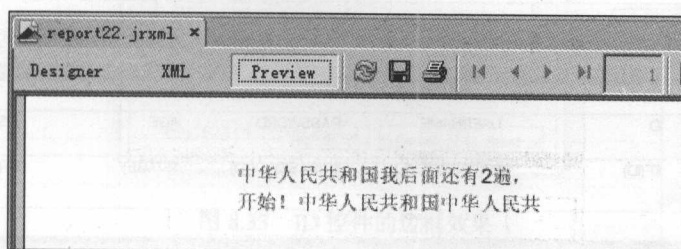


图 8.27 显示不全

实现步骤是:先创建一个变量,并设置 Variable Expression 属性值,然后在报表中显示该变量,使用 Text Field 控件显示变量的值,如图 8.28 所示。

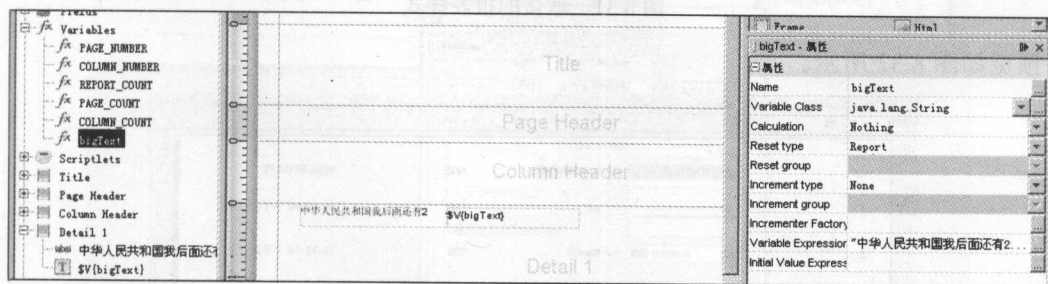


图 8.28 使用 Text Field 控件显示变量的值

最为关键的是要设置 Text Field 控件的属性值为真,如图 8.29 所示为设置 Stretch With Overflow 值为 true。

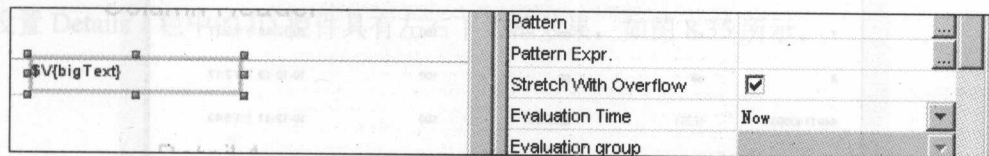


图 8.29 设置 Stretch With Overflow 值为 true

完整显示的运行效果如图 8.30 所示。

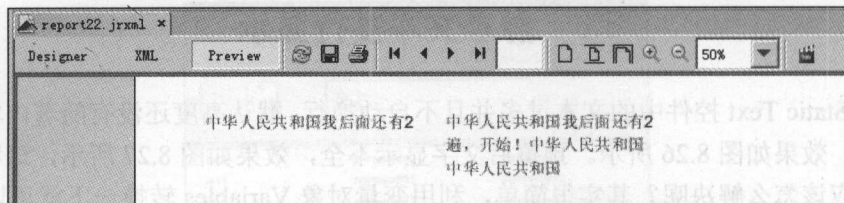


图 8.30 完整显示

8.7 设计带边框的表格

初始设计报表模板的默认样式如图 8.31 所示。

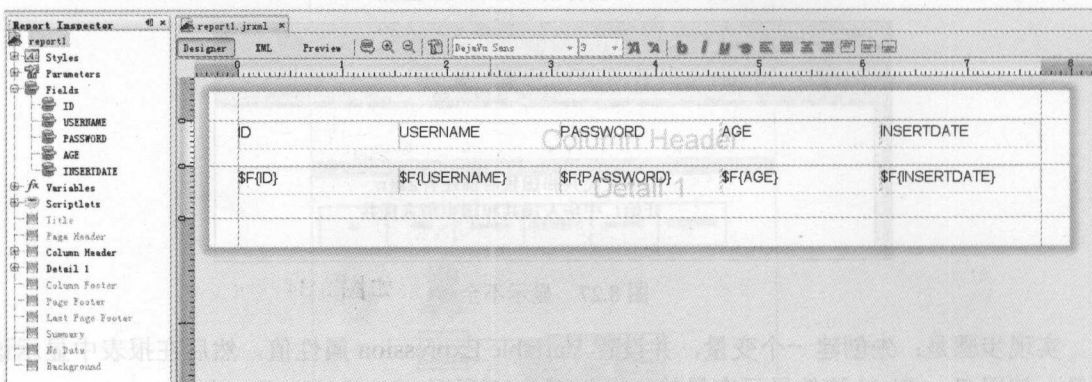


图 8.31 默认的报表样式

预览如图 8.32 所示。

ID	USERNAME	PASSWORD	AGE	INSERTDATE
1	a1	aa	100	10-12-19 下午2:17
2	a2	aa	100	10-12-19 下午2:17
3	a3	aa	100	10-12-19 下午2:17
4	a4	aa	100	10-12-19 下午2:17
5	a5	aa	100	10-12-19 下午2:17
6	a6	aa	100	10-12-19 下午2:17
7	a7	aa	100	10-12-19 下午2:17
8	a8	aa	100	10-12-19 下午2:17
44911400001	法国1	法国人 11	100	10-12-31 上午9:43
44911410001	法国2	法国人 22	200	10-12-31 上午9:43

图 8.32 默认外观的预览效果

如果想在预览时出现表格边框的效果,可以设置表头的 ID 控件具有四周边框效果,如图 8.33 所示。

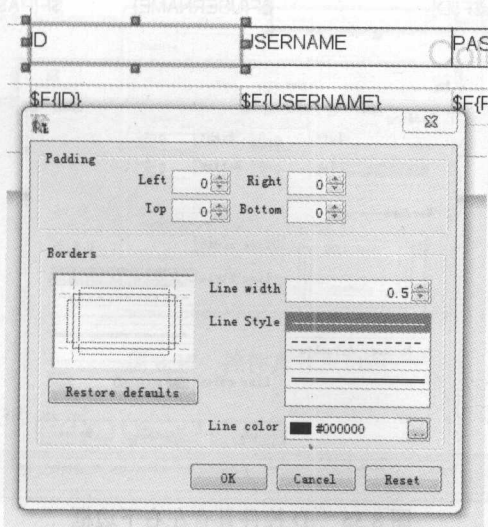


图 8.33 ID 控件的边框效果

设置其他 Column Header 中控件具有上下右边框,效果如图 8.34 所示。

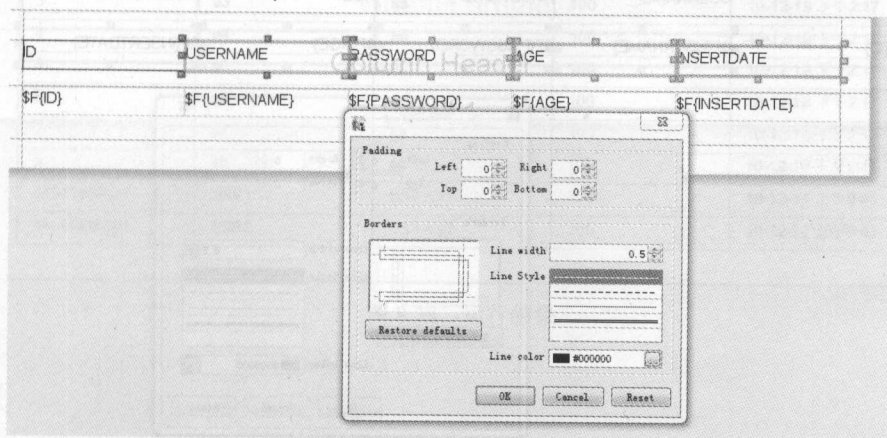


图 8.34 其他控件的边框效果

至此列头的控件设计完毕,下一步设计 Detail 1 栏中的控件。

设置 Details 1 栏中的 ID 控件具有左右下边框效果,如图 8.35 所示。

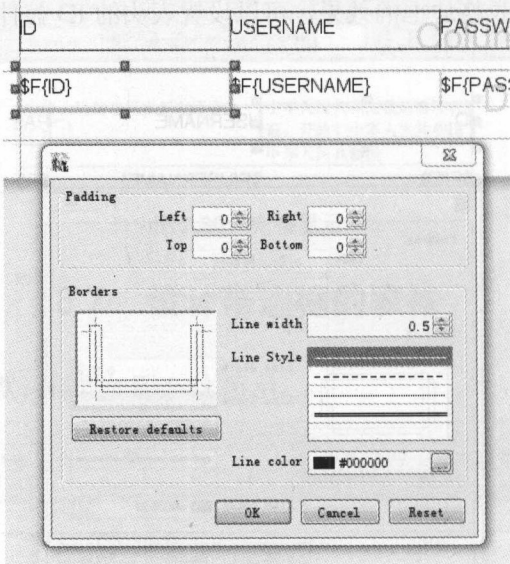


图 8.35 ID 控件具有左右下边框

其他控件具有右下边框，如图 8.36 所示。

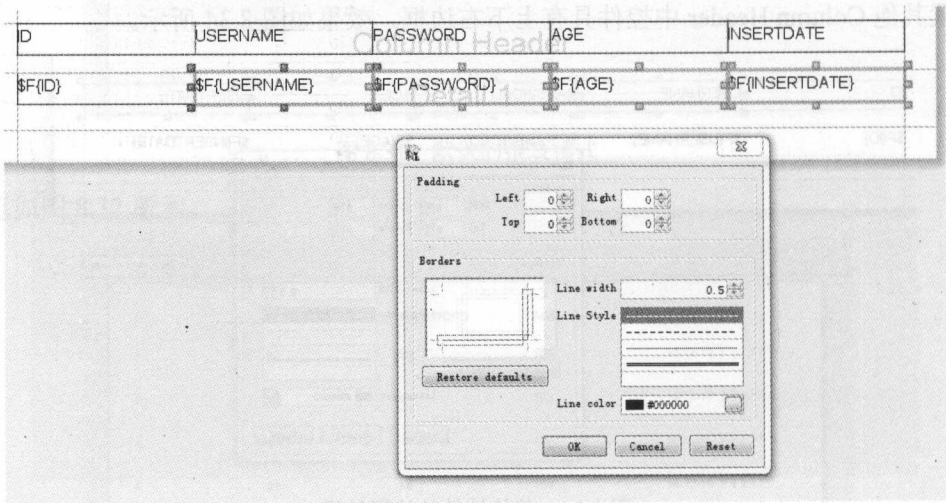


图 8.36 其他控件边框

调整列头 Column Header 的高度和 Column Header 中控件的高度，即高度相同，还要调整 Details 1 栏中控件的高度和 Details 1 栏 Band 的高度相同。

调整控件的左边距，如图 8.37 所示。

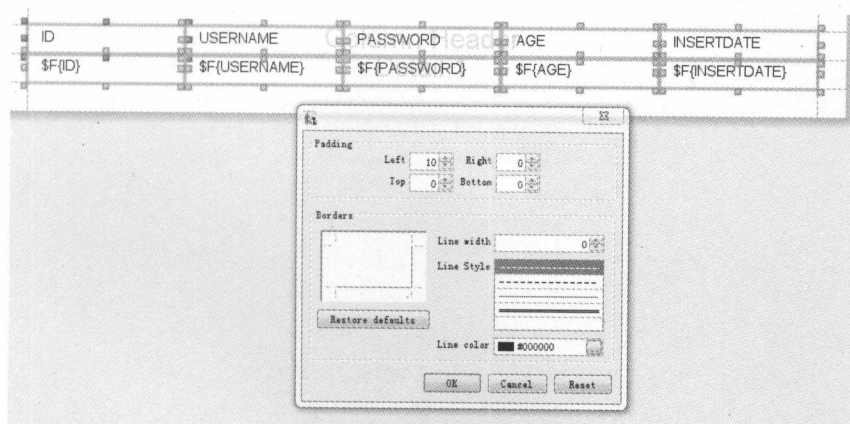


图 8.37 控件具有左边距

运行模板效果如图 8.38 所示。

ID	USERNAME	PASSWORD	AGE	INSERTDATE
1	a1	aa	100	10-12-19 下午2:17
2	a2	aa	100	10-12-19 下午2:17
3	a3	aa	100	10-12-19 下午2:17
4	a4	aa	100	10-12-19 下午2:17
5	a5	aa	100	10-12-19 下午2:17
6	a6	aa	100	10-12-19 下午2:17
7	a7	aa	100	10-12-19 下午2:17
8	a8	aa	100	10-12-19 下午2:17
44911400001	法国1	法国人11	100	10-12-31 上午9:43
44911410001	法国2	法国人22	200	10-12-31 上午9:43

图 8.38 运行结果